上海交通大學

SHANGHAI JIAO TONG UNIVERSITY

学士学位论文

THESIS OF BACHELOR

论文题目：Higher Accuracy Path Planning Algorithm Using Factored Eikonal Equation

学生姓名：戚东平

学生学号：5130519048

专　　业：致远数学班

指导教师：Alexander Vladimirsky; 张镭

学院(系)：致远学院

路径规划问题的高精度算法
—使用 Factored Eikonal Equation

摘要

Eikonal 方程来源于多种学科，如波前的传播问题，几何光学以及最优控制理论。如波前的传播，我们若考虑速度函数为各向同性，那么波前到达一个点 $\boldsymbol{x}$ 的时间函数将满足 Eikonal 方程。还有许多其它方法导出这一方程。

然而，当我们解的初值只有一个单点源时，会出现 rarefaction fan 的现象。这将会导致解的二阶导数在点源附近趋于无穷，使得数值方法的精确性受到影响。一个解决方法是使用“Lagrangian 技巧”，这种方法假设特征线是直线并通过积分得到点源附近的函数值。另一种方法是使用 factored Eikonal 方程。它将原方程分解为两部分，一部分包含奇性而另一部分很光滑。

这两种方法对单点源造成的 rarefaction fan 效果很好。但 rarefaction fan 还可能出现在其它情况中，尤其是带有障碍物的路径规划问题。如果我们能够在 rarefaction fan 出现的角点附近使用这两种方法的话，有可能帮助我们恢复一阶收敛性。

在本文中，我们将介绍一种新的方法，可以帮助恢复一阶收敛性。文章的结构是：首先，我们介绍 Eikonal 方程的背景以及粘性解理论。之后我们会介绍已有的数值方法，它们不具备一阶收敛性。接下来介绍“Lagrangian 技巧”和 factored Eikonal 方程，之后会介绍障碍造成的 rarefaction fan 的现象。最后我们提出一种新算法，并展示一些数值算例和收敛性研究。

关键词：最优控制，路径规划问题，factored Eikonal 方程

HIGHER ACCURACY PATH PLANNING ALGORITHM USING FACTORED EIKONAL EQUATION

ABSTRACT

The Eikonal Equation arises from different fields of science subjects, such as the front propagation problem, the geometric optics and the optimal control theory. Take the front propagation problem as an example: If we consider the speed function of the moving front to be isotropic, then the time function $u(\mathbf{x})$ when the front reach a specific point $\mathbf{x}$ will satisfy the Eikonal equation. There are still other ways to derive the equation.

However, the phenomenon of ‘rarefaction fan’ happens when we are solving the Eikonal equation with a point-source initial condition. This will cause the second-order partial derivatives to grow to infinity close to the source, which reduces the accuracy of numerical method. One solution to this problem is to use the “Lagrangian Technique”, which by pretending the lines of characteristics to be straight and integrate to obtain the value around the rarefaction fan. Another way is to use the factored Eikonal equation, which factors the original equation into two parts, one of them containing the singularity while the other will be quite smooth.

The “Lagrangian Technique” and the factored Eikonal equation work well for the rarefaction fan caused by an isolated initial point source. However, the rarefaction fan may appear in other cases, especially when we are dealing with path planning problems with obstacles. If we can use these two techniques to solve the Eikonal equation around the “corner points” where the rarefaction fan appears, it might recover the first order convergence.

In this article, we will introduce a new factoring method which can help us recover first-order convergence. The structure of this article will be: firstly, we will introduce the background of Eikonal equation and the theories about the viscosity solution.Then we will introduce the existing numerical method solving this equation, which cannot obtain first order accuracy. The following part contains the introduction of "Lagrangian Technique" and factored Eikonal equation. Then we will show the rarefaction fan caused by the obstacles. Finally we will describe the new algorithm and several numerical examples and convergence studies will be provided.

Keywords: optimal control, path planning problem, factored Eikonal equation

Contents

List of Figures

Chapter 1 Introduction

1.1 Optimal Control Theory

The optimal control theory is a branch of the calculus of variations, dealing with the problem of finding a control policy for a given system such that a certain optimal criterion is achieved [20]. An optimal control is a set of differential equations describing the paths of the control variables that minimize (or maximize) the cost function. Usually it has the following form [8]

$$\begin{cases} \mathbf{y}'(t) = f(\mathbf{y}(t), \mathbf{a}(t))\mathbf{a}(t), & t > 0 \\ \mathbf{y}(0) = x_0, & x_0 \in \Omega \end{cases} \tag{1-1}$$

- Control Parameter: $\mathbf{a}(t)$
- Time-to-boundary: $T(x, \mathbf{a}(\cdot)) = \inf\{t \in \mathbb{R}_+ | y(t) \in \partial\Omega\}$
- Exit Time-penalty: $q : \partial\Omega \mapsto \mathbb{R}_+$
- Cost Function: $\mathrm{Cost}(x, \mathbf{a}(\cdot)) = \int_0^T K(\mathbf{y}(t), \mathbf{a}(t))\, dt + q(\mathbf{y}(T))$
- Value Function: $u(x) = \inf_{\mathbf{a}(\cdot)\in\mathcal{A}}\{\mathrm{Cost}(x, \mathbf{a}(\cdot))\}$

This is the general structure of a problem in optimal control theory. One of the most useful result of the optimal control theory is the Eikonal equation, which will be later discussed.

1.2 Eikonal Equation

The Eikonal equation arises from different fields of mathematics and physics, such as the wave propagation problem, the geometric optics and optimal control theory [19]. One of its most famous derivation is through the theory of optimal control. It is a first order non-linear partial differential equation of the form

$$\begin{cases} \|\nabla u(\mathbf{x})\| f(\mathbf{x}) = 1, & \mathbf{x} \in \Omega \\ u(\mathbf{x}) = g(\mathbf{x}), & \mathbf{x} \in \Omega_0 \end{cases} \tag{1-2}$$

where Ω_0 is the set of initial conditions. It is a special case of the Hamilton-Jacobi-Bellman Equation, which is the fundamental equation in optimal control theory as a result of Bellman's Optimal Principle. It is well known that the theory of viscosity solution will guarantee the existence and uniqueness of the solution of Hamilton-Jacobi-Bellman equation [7]. Therefore as a special case, the Eikonal equation has a unique solution under this meaning.

1.3 Rarefaction Fan

Consider the simple case where $f(\mathbf{x}) = 1$, $u = 0$ on Ω_0 and Ω_0 contains only one point x_0, then $u(\mathbf{x})$ will give us the distance function from each point to the initial set Ω_0. This means that there are a bunch of characteristics spray from x_0. This phenomenon is called the "rarefaction fan". As long as the initial set is just a point, there will be a rarefaction fan even if $f(\mathbf{x})$ is not constant.

The rarefaction fan comes from the hyperbolic conservation law, where it is called the rarefaction wave, comparied with the shock wave [8]. The rarefaction happens when a series of characteristics emitting from the same point while the shock wave is due to the characteristics collide with each other.

1.4 Fast Marching Method

As for the discrete shortest path problem defined on graph, there already exists several efficient method, such as the Bellman-Ford algorithm and the Dijkstra's algorithm [4]. However, there is a hugh jump between the discrete problem and the continuous method, which means sometimes just by refining the grid size we cannot obtain the true solution in the limit. Therefore, we need to develop new numerical schemes.

There are many numerical methods for Eikonal equation, such as the Fast Marching

Method [19] [17], which is an analogue of the Dijkstra's method. The method is first-order, based on a Godunov-type monotone upwinding scheme. However, the method actually cannot have first-order accuracy if there appears a rarefaction fan. Because the second derivatives of the value function will become unbounded near the source, which degrades the accuracy of the numerical method. As a result, we need some further techniques if we want to obtain first order convergence.

1.5 Solution Techniques

In order to circumvent this, two techniques might be helpful. One is to use the "Lagrangian technique", which integrate along straight lines to obtain the value on each point around the rarefaction fan. The other is to use a different "factored Eikonal equation", where the original value function is factored into either a product or a sum of two functions, one of them containing the singularity [13]. Then the original PDE will be replaced by a PDE with respect to a new function, which does not have singularity and is quite smooth.

However, the rarefaction can not only be caused by an isolated initial point source, but also cases like the appearance of obstacles. In order to recover the first order accuracy in the path planning problems with obstacles, we intend to use these two techniques to improve the accuracy of the original Fast Marching Method. In particular, the method can be applied to the maze navigation problem where there are many obstacles and many rarefaction fans.

1.6 Structure of Paper

The paper is organized as follows. We will first introduce the background knowledge of optimal control theory and some results. Then we will derive the Eikonal equation and explain how its numerical method works. Then we will bring up the phenomenon of rarefaction fan. We will show that the former method cannot achieve first-order convergence. The next part involves the idea of "Lagrangian technique" and the prior work on factored Eikonal equation. In the following section we will discuss about the rarefaction fan caused by the obstacles and then apply the two methods to deal with this problem. Several numerical examples and convergence study will be provided.

All the problems considered in this paper will be in $\mathbb{R}^2$.

Chapter 2 Optimal Control and Dynamic Programming

In this chapter, we will give a brief introduction on the background of the whole paper, which is mainly about the optimal control theory. Then we will bring up the time-optimal trajectory problem, which is very close to the path planning problem. One of the most important method in optimal control theory is the principle of dynamic programming, which will also be included in this chapter. We will introduce both the discrete and continuous version of dynamic programming principle. Finally, we will point out that under the structure of dynamic programming, the value function can be viewed as a fixed point.

2.1 Optimal Control Theory

The optimal control theory deal with the problem of finding a control law for a given system such that a certain optimal criterion is achieved [20]. An optimal control is a set of differential equations describing the paths of the control variables that minimize (or maximize) the cost function.

Remark 2.1. *People who are familiar with the classical mechanics understand that the Lagrangian mechanics takes the Principle of Least Action as the basic hypothesis and the true motion of particles will take the path with least action in the phase space [10]. The classical mechanics can actually be derived from the calculus of variations or optimal control theory, with a specific Hamiltonian or Lagrangian.*

All the optimall control theory problems have a similar structure. The form of the the basic problem in optimal control can be described as follows [8]:

$$\begin{cases} \mathbf{y}'(t) = f(\mathbf{y}(t), \mathbf{a}(t))\mathbf{a}(t), & t > 0 \\ \mathbf{y}(0) = x_0, & x_0 \in \Omega \end{cases} \tag{2-1}$$

Here $\mathbf{a}(t)$ is the control parameter, $\mathbf{y}(t)$ represent the curve or the path of our moving unit, given the initial state as x_0. The problem is defined inside the region of Ω, which is a subset of $\mathbb{R}^n$. It is possible that we add some other contraints on the motion of this unit. This means that the control parameter $\mathbf{a}(t)$ is not totally free. The set of all the measurable control parameters $\mathbf{a}(t)$ consists of a set $\mathcal{A} = \{\mathbf{a} : \mathbb{R}_+ \to \Omega |\ \mathbf{a}(\cdot)$ is measurable $\}$ called the collection of all admissible controls [8]. Whenever we are solving the problem, we need that the control parameter belongs to $\mathcal{A}$.

The criterion of the problem lies in the cost function. The problem next defines a cost function which helps us to determine what is the "optimal control" to our desire, usually minimizing or maximizing the cost function due to different purposes. The cost function is in general defined as an integral

$$C(x_0, \mathbf{a}(\cdot)) = \int_0^T K\left(\mathbf{y}(t), \mathbf{a}(t)\right) dt + q\left(\mathbf{y}(T)\right)$$

Here T is defined as the terminal time $T(x, \mathbf{a}(\cdot)) = \inf\{t \in \mathbb{R}_+ | \mathbf{y}(t) \in \partial\Omega\}$. $K\left(\mathbf{y}, \mathbf{a}\right)$ is the cost weight, which can be viewed as the traffic burden or the density of inhomogeneous materials. The function $q(\cdot)$ is the terminal payoff, which means that before running out of the domain, the unit needs to pay an extra "penalty".

According to different requirements, we might want to find out the "optimal solution" which minimizes or maximizes the cost function above while follows the dynamics in eqution (2-1). If we obtain the optimal control at each point, then we can calculate the cost value on each point, which form into a new function called the value function $u(x)$. This is the fundamental structure of a problem in the optimal control theory. The main functions and variables can be listed as follows.

- Control Parameter: $\mathbf{a}(t)$
- Time-to-boundary: $T(x, \mathbf{a}(\cdot)) = \inf\{t \in \mathbb{R}_+ | y(t) \in \partial\Omega\}$
- Exit Time-penalty: $q : \partial\Omega \mapsto \mathbb{R}_+$
- Cost Function: $\mathrm{Cost}(x, \mathbf{a}(\cdot)) = \int_0^T K\left(\mathbf{y}(t), \mathbf{a}(t)\right) dt + q\left(\mathbf{y}(T)\right)$
- Value Function: $u(x) = \inf_{\mathbf{a}(\cdot) \in \mathcal{A}}\{\mathrm{Cost}(x, \mathbf{a}(\cdot))\}$

2.2 Time-optimal Trajectory Problem

In this section, we will consider a simpler optimal control problem on $\mathbb{R}^2$, which is quite similar to the "shortest path problem". The optimal policy in the optimal control theory need not to be real "paths". Sometime they can be a sequence of operations or decisions, or some "orbits" in higher dimension spaces. However, in this problem, we actually are seeking for the paths.

The control parameter will become simpler here. It will be restricted to only about the directions. The problem has the following form [18], compared to 2-1:

$$\begin{cases} \mathbf{y}'(t) = f(\mathbf{y}(t), \mathbf{a}(t))\mathbf{a}(t), & \|\mathbf{a}(t)\| = 1, \\ \mathbf{y}(0) = x_0, & x_0 \in \Omega \subseteq \mathbb{R}^2 \end{cases} \tag{2-2}$$

Here $f : \mathbb{R}^2 \times S^1 \mapsto \mathbb{R}_+$ is the scalar speed function which describes how quick can the unit moves at each specific point. The speed function is always positive, which means that there are no "stopping points". The control parameter $\mathbf{a}(t)$ is now defined only on the unit circle, which means that at each point we only need to determine "to which direction" we want the unit to go.

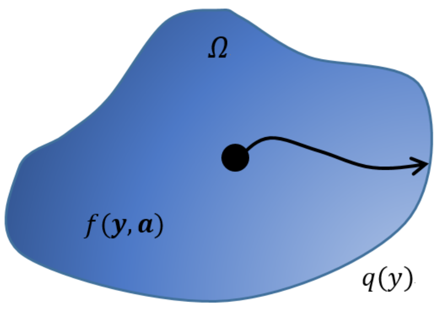


Figure 2.1 Time-optimal trajectory problem

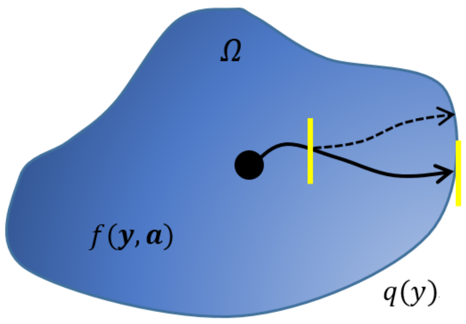
$$C(x_0, \mathbf{a}(\cdot)) = T(x_0, \mathbf{a}(\cdot)) + q\left(\mathbf{y}(T)\right)$$

The cost function here is defined to be only the time a unit need to reach the boundary and the terminal penalty. This is equal to letting the cost weight to be constantly one everywhere, $K(\cdot,\cdot) = 1$. So we call this the “time-optimal” trajectory problem.

2.3 Dynamic Programming

Dynamic programming is usually referred to as a method in algorithm theory which solves a problem by breaking it into several subproblems that have similar properties [6]. A dynamic programming algorithm solves each sub-subproblems just once and then saves its answer in a table, thereby avoiding the work of recomputing the answer every time it solves each sub-subproblem. If subproblems can be nested recursively inside larger problems, so that dynamic programming methods are applicable, then there is a relation between the value of the larger problem and the values of the sub-problems. The main idea of dynamic programming can be interpreted by the following :

Theorem 2.2. *(Principle of Optimality) [2] An optimal policy has the property that whatever the initial state and initial decision are, the remaining decisions must constitute an optimal policy with regard to the state resulting from the first decision.*



Sub-strategy is optimal!

Figure 2.2 Bellman optimal principle

This optimal principle can be simply proved by the following philosophy: if the remaining policies π_1 are not optimal, then there must be another policy $\tilde{\pi_1}$ to be the optimal. The former policy combined with this optimal sub-policy$\tilde{\pi_1}$, which forms into a new optimal policy, is better than the original one. This contradicts to the optimality of the original policy. As a result, all the sub-policies of an optimal policy should also be optimal. (This can be viewed as another expression of this principle)

There are several different versions of Bellman's principle of optimality, such as the discrete version, the continuous version and the stochastic version [3]. All the descriptions can be written into an equation called the Bellman equation, which shows the mathematical structure of Bellman's principle of optimality. The Bellman equation is not an analytic expression of the solution, however, it is a necessary condition of the problem. In the next section, we will focus on two different versions of Bellman equation: the discrete and continuous cases.

2.4 Bellman Equation

2.4.1 The Discrete Bellman Equation

The discrete version of Bellman equation usually appears in the discrete version of shortest path problem [3]. The problem is defined on a graph and there are different weights on each edge $K(Y_i, Y_j)$ which can be considered as the traffic burden on a road, or the length between different cities. Now the control parameter need to determine a sequence of points as the path of each unit. The time-to-boundary now becomes the steps one unit need to reach the boundary. All the integrals in the former problem will become summations here. The relevant parameters will be listed as follows.

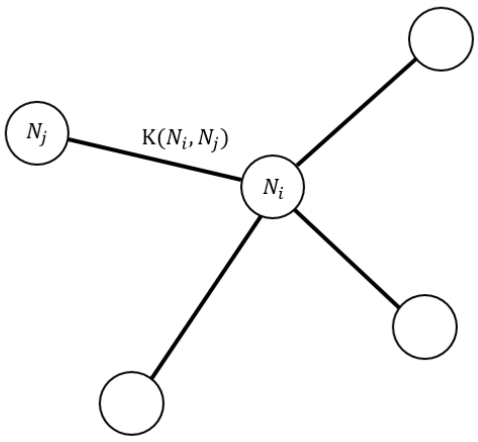


Figure 2.3 Discrete shortest path problem

- Control Parameter: $\mathbf{Y} = \{Y_1, Y_2, \cdots, Y_T\}$

- Exit Time-penalty: $q : \partial G \mapsto \mathbb{R}_+$
- Cost Function: $\mathrm{Cost}(Y_0, \mathbf{Y}) = \sum_{i=0}^{T} K(Y_i, Y_{i+1}) + q(Y_T)$
- Value Function: $u(Y_0) = \min_{\mathbf{Y}\in\mathcal{Y}}\{\mathrm{Cost}(Y_0, \mathbf{Y})\}$

Suppose that we know the optimal policy for one point N_i is $N_i, N_i^1, N_i^2 \cdots$, then according to Bellman's principle of optimality, the value on each point among this policy sequence should be the sum of the value of its former neighbor and the edge weight between them. Therefore, due to the optimality of the value function, we have the following discrete version Bellman's equation [3]:

$$\begin{cases} u(Y_i) = \min\limits_{Y_j \in N(Y_i)} \{K(Y_i, Y_j) + u(Y_j)\}, & \text{if } i < T, \\ u(Y_i) = q(Y_i), & \text{if } i = T \end{cases} \tag{2-3}$$

2.4.2 The Continuous Bellman Equation

The continuous version of Bellman equation is just the same as the discrete one, while there are few things need to be changed. The first is since we have a continuous domain now, the minimizing operator should be changed into the infimum, since we do not know if we can reach that infimum. However, the control policy now cannot by described simply be a sequence of points. Instead, there are an uncountablly infinite number of possible policies, therefore, the admissible control set is more complicated.

A similar argument as the former one helps give us the following continuous version of Bellman equation [12]:

$$\begin{cases} u(\mathbf{x}) = \inf\limits_{\mathbf{a}(\cdot)\in\mathcal{A}} \left\{ \int_0^{\tau} K\left(\mathbf{y}(s), \mathbf{a}(s)\right) + u\left(\mathbf{y}(\tau)\right) \right\}, & \text{if } \mathbf{x} \in \Omega \\ u(\mathbf{x}) = q(\mathbf{x}), & \text{if } \mathbf{x} \in \partial\Omega \end{cases} \tag{2-4}$$

Here τ is a time step small enough to make sure that $\mathbf{y}(\tau)$ still remains inside the domain. $\mathbf{y}(\tau)$ satisfies $\mathbf{y}(0) = \mathbf{x}$ If we consider the Time-optimal trajectory problem, then the corresponding equation will be

$$\begin{cases} u(\mathbf{x}) = \inf\limits_{\mathbf{a}(\cdot)\in\mathcal{A}} \left\{\tau + u\left(\mathbf{y}(\tau)\right)\right\}, & \text{if } \mathbf{x}\in\Omega \\ u(\mathbf{x}) = q(\mathbf{x}), & \text{if } \mathbf{x}\in\partial\Omega \end{cases} \tag{2-5}$$

We need to notice that Bellman equation does not help us solve the system explicitly, it is another way of explaining the properties of the value function $u(x)$. Although this is not a solution, this illustrates a very good property of the value function, which can later help us obtain more results.

Remark 2.3. *There are basically two ways of researching on the optimal control theory: the Dynamic Programming Pringciple and the Maximum Principle, developed by the Russian mathematician Lev Pontryagin and his student [14]. It involves several state equations describing the motion of the unit and some maximum conditions.We will not introduce too much on this principle here.*

2.5 Value Function as Fixed Point

One thing we need to notice is that, in the Bellman equation if we regard the minimum or infimum function as operators on the value function itself, then $u(\mathbf{x})$ can be seen as a fixed point of these operators [3].

$$u = \mathcal{F}(u)$$

$$\begin{cases} u(\mathbf{x}) = \inf\limits_{\mathbf{a}(\cdot)\in\mathcal{A}} \left\{\tau + u\left(\mathbf{y}(\tau)\right)\right\}, & \text{if } \mathbf{x}\in\Omega \\ u(\mathbf{x}) = q(\mathbf{x}), & \text{if } \mathbf{x}\in\partial\Omega \end{cases} \tag{2-6}$$

Due to the results in functional analysis, we have the fixed point theorem or contraction mapping theorem as tools, which can help us develop existence and uniqueness on the solution to specific partial differential equations. There are some interesting results developed in [5].

The following is a most common version of contraction mapping theorem.

Theorem 2.4. (Contraction mapping theorem) *[16]*

If X is a complete metric space, and if ϕ is a contraction mapping of X into X, that is there exists a number $0 < c < 1$ such that $d(\phi(x), \phi(y)) \leq d(x, y)$, then there exists one and only on $x \in X$ such that $\phi(x) = x$.

Later we will see that in this point of view, we can easily obtain some iterative numerical method both in the discrete case on graph or continuous case in $\mathbb{R}^n$.

2.6 Conclusion

In this chapter, we have introduced the background of optimal control theory and time-optimal trajectory problem. Besides, the Bellman's optimal principle gives us a very helpful way to understand the solution. This shows us that there is some special structures within the theory of optimal trajectory problems. In the last part we talked about viewing the solution as a fixed point. Later we will see that this problem has a very important use.

Chapter 3 Hamilton-Jacobi-Bellman Equation and Viscosity Solution

Although Bellman's principle of optimality points out a very important property of the value function $u(\mathbf{x})$, it does not help us solve the problem or explicitly give us the analytic solution. However, if we suppose $u(\mathbf{x})$ has some smoothness, then we can change the form of Bellman's equation and finally obtain a necessary condition on the principle: a partial differential equation. This equation is called the Hamilton-Jacobi-Bellman equation and is widely used in the optimal control. Its derivation is simple and here we only give a brief outline.

3.1 Derivation of HJB equation

We will use (2-4) to derive the general form of the Hamilton-Jacobi-Bellman equation (HJB equaiton). Now we list down some hypotheses on the variables and functions [18].

- $u(\mathbf{x})$ is at least first-order continuously differentiable.
- $K(\cdot,\cdot)$ is Lipschitz continuous and $0 < K_1 \leq K(x,\mathbf{a}) \leq K_2 < \infty$.
- $0 < q_1 \leq q(\mathbf{x}) \leq q_2 < \infty$.
- $\mathbf{y}(\cdot)$ and $\mathbf{a}(\cdot)$ are both differentiably dependent on time.

The first hypothesis on $u(\mathbf{x})$ helps us use the Taylor expansion on its variables. $K(\cdot,\cdot)$ to be Lipschitz continuous is reasonable, since usually the cost between two close points is proportionate to the distance between them. The fact that $K(\cdot,\cdot)$ is bounded by some lower positive value is quite important for later we will give some results of causality using this condition. The last hypothesis about the differentiably dependence is also reasonable.

If we require τ to be small enough then we can get

$$\begin{aligned} \mathbf{y}(\tau) &= \mathbf{y}(0) + \mathbf{y}'(0)\tau + O(\tau^2) \\ &= \mathbf{x} + \tau f\left(\mathbf{x}, \mathbf{a}(0)\right)(0)\mathbf{a}(0) + O(\tau^2) \end{aligned} \tag{3-1}$$

Next, due to the hypotheses made above, we can make a Taylor expansion on $u(\mathbf{x})$ and a first order approximation on $K(\mathbf{x}, \mathbf{a})$ (due to Lipschitz continuity), then we can obtain [18]:

$$\begin{aligned} u(\mathbf{x}) &= \inf_{\mathbf{a}(\cdot)\in\mathcal{A}} \left\{ \int_0^\tau K\left(\mathbf{y}(s), \mathbf{a}(s)\right) + u\left(\mathbf{y}(\tau)\right) \right\} \\ &= \inf_{\mathbf{a}(\cdot)\in\mathcal{A}} \left\{ \int_0^\tau K\left(\mathbf{y}(0) + O(\tau), \mathbf{a}(0) + O(\tau)\right) + u\left(\mathbf{y}(0) + \mathbf{y}'(0)\tau + O(\tau^2)\right) \right\} \\ &= \inf_{\mathbf{a}(\cdot)\in\mathcal{A}} \left\{ K\left(\mathbf{y}(0), \mathbf{a}(0)\right) + u\left(\mathbf{y}(0)\right) + \tau f\left(\mathbf{x}, \mathbf{a}(0)\right) \nabla u\left(\mathbf{y}(0)\right) + O(\tau^2) \right\} \\ &= \tau K\left(\mathbf{x}, \mathbf{a}(0)\right) + u(\mathbf{x}) + O(\tau^2) + \tau \inf_{\mathbf{a}(\cdot)\in\mathcal{A}} \left\{ f\left(\mathbf{x}, \mathbf{a}(0)\right) \nabla u\left(\mathbf{x}\right) \right\} \end{aligned} \tag{3-2}$$

Since our set of admissible policies $\mathcal{A}$ is defined on the unit circle S^1, which is a compact set in $\mathbb{R}^2$, therefore the infimum of the variables inside the brace can be obtained. Then we can change the infimum into minimum. By eliminating the $u(\mathbf{x})$ on both sides and dividing by τ we can obtain [8]:

$$\min_{\mathbf{a}(\cdot)\in S^1} \left\{ f\left(\mathbf{x}, \mathbf{a}\right) \mathbf{a} \cdot \nabla u(\mathbf{x}) \right\} + K(\mathbf{x}, \mathbf{a}) = 0 \tag{3-3}$$

Equation (3-2) is called the Hamilton-Jacobi-Bellman equation.

3.2 Eikonal Equation

In general, the speed function $f(\mathbf{x}, \mathbf{a})$ is not independent of the control parameter $\mathbf{a}$, which in this case is the direction. However, if we consider the simple case where the speed function is only relevant with the position variable $\mathbf{x}$, i.e. $f = f(\mathbf{x})$ then we can simplify the form of Hamilton-Jacobi-Bellman equation.

$$\begin{aligned}\min_{\mathbf{a}(\cdot)\in S^1}\left\{f(\mathbf{x})\mathbf{a}\cdot\nabla u(\mathbf{x})\right\}+K(\mathbf{x},\mathbf{a})=0\\ f(\mathbf{x})\min_{\mathbf{a}(\cdot)\in S^1}\left\{\mathbf{a}\cdot\nabla u(\mathbf{x})\right\}+K(\mathbf{x},\mathbf{a})=0\\ -f(\mathbf{x})\|\nabla u(\mathbf{x})\|+K(\mathbf{x},\mathbf{a})=0\end{aligned} \tag{3-4}$$

In the case of Time-optimal trajectory problem, the cost weight is constantly one, i.e. $K(\mathbf{x},\mathbf{a})=1$, this leads to the final form of Eikonal equation:

$$\|\nabla u(\mathbf{x})\|f(\mathbf{x})=1 \tag{3-5}$$

The Eikonal equation arises from the wave propagation problem and geometric optics. It has a wide range of applications from path planning to isotropic optimal control or isotropic front propagation [19]. In particular, if the speed function $f(\mathbf{x})=1$, then the solution $u(\mathbf{x})$ is actually the distance function since if the speed everywhere is constantly one, then the travel time will be equal to the distance.

3.3 Viscosity Solution

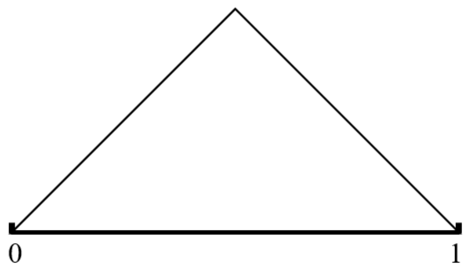
The Eikonal equation is a first-order nonlinear partial differential equation. Usually we suppose that the speed function $f(\mathbf{x})$ is Lipschitz continuous. Then there arises the question that if we give some initial conditions with good property, then what can we predict on the existence and uniqueness of the solution?

The modern partial differential equation theory no longer tries to find the analytic solution for all the equations, Instead, people are more interested in the function spaces where the solutions lie in. The weak formula of special PDEs help us form the idea of weak solution and weak derivatives. The weak solutions are almost everywhere differentiable so there can exist a set of non-differentiable points with zero measure. This means that there can exist even countably mant points where the derivatives here are not continuous. Consider the following simple one-dimensional Eikonal equation where $f(x) = 1$:

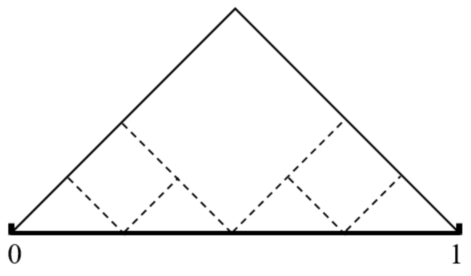
$$\begin{cases} |u'(x)| = 1, x \in [0, 1] \\ u(0) = 0, u(1) = 0 \end{cases} \tag{3-6}$$

Since it is a one-dimensional problem, the gradient becomes just derivative. It is quite obvious that we cannot find a smooth solution since $u'(x) = \pm 1$, and if we suppose that $u(x)$ is smooth, then $u'(x)$ should be a constant in the whole interval. However, this will contradict to the boundary condition. (By Rolle Theorem there should be a point where $u'(x) = 0$)

However, this problem has a very clear physical meaning. The distance function in 3.1(a) can be viewed as a solution while there is a non-differentiable point at 0.5. In this sence, the distance function is a weak solution.



(a) distance function



(b) other solutions

Figure 3.1 Infinite weak solutions

However if we add more non-differentiable points inside this interval, that is adding more points where the derivative jumps from -1 to 1, then we can define more solutions which satisfies the equation while having finitely many non-differentiable points. These solutions lool like a saw or an indented line. Therefore, there will be infinitely many weak solutions that satisfied the equation. The uniqueness is lost in the classical theory of weak solution.

Since only the distance function is what we really want, we need to define a new type of solution which can rule out the other choices. Defined by Pierre-Louis Lions, Michael G. Crandall and Lawrence Evans, the theory of viscosity solution offers a satisfactory answer [7].

According to the paper of Pierre-Louis Lions and Michael Crandall in 1983 [7], they defined a new type of solution for equation 3-7 as follows:

Definition 3.1. (Crandall, Lions, 1983) *[7] v is a viscosity solution for equation 3-7 if for every function $\phi \in C^1(\Omega)$, v satifies*

- *viscosity subsolution: if $v - \phi$ has a local maximum at $x_0 \in \Omega$ then,*

$$\min_{\boldsymbol{a}\in A}\{(\nabla\phi(x_0)\cdot\boldsymbol{a})f(x_0,\boldsymbol{a})\} + K(x_0,\boldsymbol{a}) \geq 0$$

- *viscosity supsolution: if $v - \phi$ has a local minimum at $x_0 \in \Omega$ then,*

$$\min_{\boldsymbol{a}\in A}\{(\nabla\phi(x_0)\cdot\boldsymbol{a})f(x_0,\boldsymbol{a})\} + K(x_0,\boldsymbol{a}) \leq 0$$

There are actually many different but equivalent ways of defining the viscosity soltuion. The above is one of them using the subsolution and supsolution. There is another way of using the set of subdifferentials and the set of supdifferentials. [9]

In the same paper, Crandall and Lions have also proved the existence and uniqueness of the solution defined as above for the Dirichlet and Cauchy problem of equation 3-7. Due to this definition, a viscosity solution to a Hamilton-Jacobi equation cannot have any points that is a local minimum, which actually helps us rule out the other solutions.

Theorem 3.2. (Crandall, Lions, 1983) *[7] Under the meaning of viscosity solution, the Hamilton-Jacobi-Bellman equation 3-7 has one unique solution.*

Remark. The name "viscosity solution" actually comes from the vanishing viscosity method [18]. If we consider the solution of the following problem

$$H\left(\mathbf{x}, u(\mathbf{x}), \nabla u(\mathbf{x})\right) + K(\mathbf{x}x, \mathbf{a}) = \epsilon\triangle u_\epsilon(\mathbf{x}) \tag{3-7}$$

Here $H\left(\mathbf{x}, u(\mathbf{x}), \nabla u(\mathbf{x})\right)$ is called the Hamiltonian, which is nothing but

$$H\left(\mathbf{x}, u(\mathbf{x}), \nabla u(\mathbf{x})\right) = \min_{\mathbf{a}(\cdot)\in S^1} \left\{ f\left(\mathbf{x}, \mathbf{a}\right) \mathbf{a} \cdot \nabla u(\mathbf{x}) \right\}$$

Since the Laplacian has very good smoothing property, then the existence of solution of the above equation becomes easy. Then we let the "viscosity term" $\epsilon \triangle u_\epsilon(\mathbf{x})$ to vanish, which means

$$u(\mathbf{x}) = \lim_{\epsilon \to 0} u_\epsilon(\mathbf{x})$$

The viscosity solution, even though in the definition there is no evidence for "viscostiy", can also be derived by this method. However, the definition we use in this section is easier to handle.

3.4 Conclusion

In this chapter we have derived the general form of the Hamilton-Jacobi-Bellman equation, which plays a key role in the optimal control theory. From there, we derive the Eikonal equation, which is a special case when the speed function is homogeneous and is not relevant to the control parameter. Then we show that under the classical meaning of the weak solutions, we might lose the uniqueness on the solution, which leads to the idea of viscosity solution. We introduce the viscosity solution, its definition and properties. The mathematical background has been set up.

Chapter 4 Fast Marching Method

In this chapter we will introduce the numerical method of solving the Eikonal equation. The iterative method came out first in the history, which is not complicated but the time consumption is huge. In 1995, James Sethian brought up a fast method, called the Fast Marching Method [17]. This is a continuous analogue of the Dijkstra's method which solves the shortest path problem on graphs.

First, we will talk about the iterative method, then introduce several classical methods of solving the shortest path problem on graphs. We are most interested in the Dijkstra's algorithm, which can help us bring up the Fast Marching Method. We will introduce the FMM in details and explain how it works. Several numerical examples of different inhomogeneoous speed functions will be provided.

4.1 Iterative Method

The Contraction Mapping Theorem in functional analysis provides us with a very general method when considering solving the partial differential equations. We can always consider one side of the equation as some operators acting on the unknown function, and then use the Fixed Point Theorem or Contraction Mapping Theorem to acquire the existence of the solution.

This idea can easily be extended to the numerical method of PDEs, since we can properly discretize the equation, construct the iterative numerical schemes and just iterate until the difference between two outputs becomes small enough. There already exist some iterative methods for the numerical method of Hamilton-Jacobi equation, as an application of shape form shading [15]. Now we consider the same thing for the Eikonal equation.

Firstly we need to discretize the value function on some grids. Suppose the value of $u(\mathbf{x})$ on discrete points are $U = (U_1, \cdots, U_n)^T$, then the discrete version of the Bellman's Optimal Principle can be written as

$$U = \mathcal{F}(U)$$

$$U_i = \mathcal{F}_i(U) = \begin{cases} \min\limits_{x_j \in N(x_i)} \{K(x_i, x_j) + U_j\}, & x_i \notin \partial\Omega \\ q(x_i), & x_i \in \partial\Omega \end{cases} \tag{4-1}$$

Notice that this operator is non-linear and fully coupled (since each point depends on all its neighbors and at each point we need a minimization on all the neighbors). Therefore the iterative method needs $O(n)$ computation in each iteration if n is the total number of points. Since on a graph each shortest path has length of steps less than n, therefore the total time consumption will be $O(n^2)$.

However, we do want to develop fast method for Eikonal equation. The next section shows us that there are "causality" within the structure of shortest path problem and we can use it to develop the Fast Marching Method.

4.2 Dijkstra's Algorithm

In network optimization, the Dijkstra's algorithm is quite famous. It helps solve the shortest path problem on a graph. Its time complexity is $O(n\text{log}n)$.

In the theory of linear algebra, we know how to diagonalize special matrices. This has a very important application in the numerical analysis, since if we can diagonalize a matrix A before solving the equation $Ax = b$, we can reduce the computation from $O(n^3)$ (Gauss elimination) to $O(n)$. Although the complexity of diagonalizing the matrix will be the same as Gauss elimination, it is still useful since we know the diagonalization of special matrices a priori, such as the five point difference scheme of Laplacian operator.

However, for non-linear problems, there are no matrix theory helping us decoupling the system. Fortunately for the shortest path problem, the physics meaning of causality is quite clear and we can use this to help us decouple the system.

4.2.1 Generic Shortest Path Algorithm

The generic shorstest path algorithm goes one step further than the iterative method. We can view each iteration as updating values on each point, therefore the points who have been updated make up a set. We call it the candidate set [4]. Due to the existence of shortest path on a graph, the value updated will be stable after a few steps of updating. Then we can move these points outside the candidate set and keep focus on the points whose values are still changing.

However, we do not know when the value of a point become stable, therefore the generic shortest path algorithm just push a point outside the candidate set after it has been updated. This process can be accelerated since if we choose a "smart way" to maintain the candidate set, then it is possible that we reduce many unnecessary updating steps.

4.2.2 Bellman-Ford Algorithm

The collection of Label Correction Methods uses a simple data structure: queue. For example the Bellman-Ford Method uses a "first in first out" rule [4] to update the queue and maintain the candidate set. This method actually performs well in many cases. The next section about causality tells us another way to update.

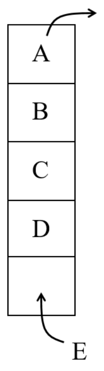


Figure 4.1 First in first out rule

4.2.3 Dijkstra's Algorithm and Causality

The fact that the speed function $f(\mathbf{x})$ is everywhere positive leads to the result that on each shortest path (or in the language of optimal control, the optimal policy), the value function will be monotonically increasing. As a consequence, if we can somehow know the shortest paths a priori, then we can update the value functions along these paths to get the exact value on them.

This is actually the decoupling of the system since we now know the relationship between points. Any points whose values are dependent on others are called coupled. In graph theory, we can use an incidence matrix to show this relationship since each two points who are connected will have an element "1" and a "0" for those who are not.

However, we do not know what is the shortest path until we have solved the problem. Therefore, we need to do this decoupling as we are solving the problem.

Firstly, the following fact of causality is obvious, it is only a result of the speed function being positive

Proposition 4.1. *[18] Suppose that on the same shortest path, if the value function on two points x_i, x_j satisfies $U_i < U_j$, then x_i will be removed from the candidate set before x_j.*

Proof. Suppose this is not true, i.e. x_j is removed from the candidate set before x_i, then by the time x_j is removed, the value U_i is still not true (larger than the true value). However since x_i is on the shortest path of x_j, therefore the value U_j is also not true, which contradicts to the fact that x_j's value become stable. □

The next proposition plays a key role in the Dijkstra's algorithm, it tells us that the point with the smallest value in the candidate set is special.

Proposition 4.2. *The point x_m with the smallest value function in the candidate set is stable (any updating will not change its value).*

Proof. The claim is equivalent to that for x_m, its former neighbor x_{m-1} on the shortest path has already become stable. For if the value of x_m is still changing by some following updating, then at least one of the points on its shortest path is still not stable, which means that its former neighbor on the shortest path is still in the candidate set.

Now we need to prove the above claim. It is quite obvious since if x_m's former neighbor on the shortest path x_{m-1} is still inside the candidate set, then the value of x_{m-1} is smaller than x_m since it is one step forward than x_m on the same shortest path, which contradicts with the minimal property of x_m. □

The above proposition shows that for each time we update some points, we can at least drag one point out of the candidate set. To find the minimum point in the candidate set is quite tricky. The usual way is to use a minimum heap which uses a binary heap to sort the list and find out the minimum point [4]. Since the time complexity of minimum heap is $O(\log n)$, the total time complexity of Dijkstra's method will be $O(n\log n)$.

The general structure of Dijkstra's method can be written as follows.

Dijkstra' Algorithm

```
Input: graph $G$, weight function $K(x_i, x_j)$, initial set $Q$

1. Add all the neighbors of the initial set $Q$ into the candidate set;
2. Check if the candidate set is empty;
3. If not, then extract the smallest $x_i$ in heap;
4. For all the neighbors $\tilde{x}_i$ of $x_i$, update its value using Bellman's optimal principle;
5. Go back to step 2.
```

The following figures can give us a simple illustration of how the algorithm are working on a graph. We can see that the candidate set pushes out one point during one updating and then includes the neighbors of all the points in the set. The one point it pushes out has the minimal value function in the candidate set. All the points in the candidate set are placed in a binary heap to help sort out the smallest point. This process is continued until the candidate set is empty.

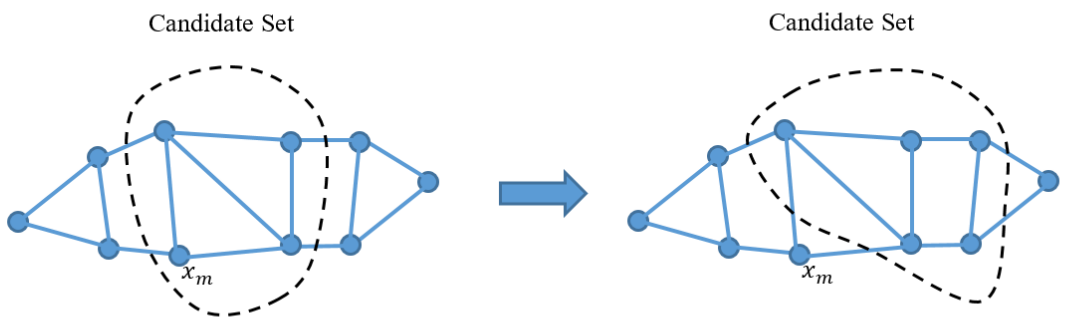


Figure 4.2 Dijkstra's algorithm

4.2.4 Discrete vs. Continuous

Although Dijkstra's algorithm works very well for the shortest path problem on graph, unfortunately it cannot be directly applied to the continuous path planning problem. The following case shows us that there is an essential difference between discrete case and continuous case.

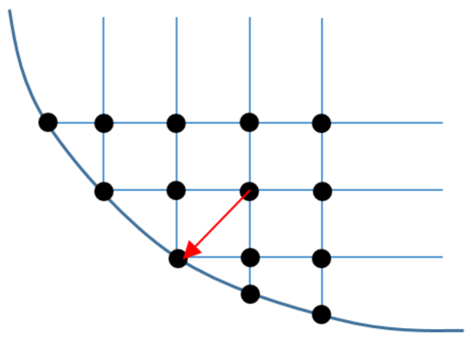


Figure 4.3 Dijkstra's failure

If we create a Cartisian grid on the above domain and use the Dijkstra's algorithm directly, for the point shown in the figure, the shortest path should be the diagonal line. However, Dijkstra's solution does not include this direction. This drawback even cannot be remedied by adding the diagonal, for there are infinitly many directions that should be included.

Another example will be more obvious. The Dijkstra's method cannot even get the true answer even if the domain is rectangular. Consider the following problem with one single sourse as the initial condition. The orange line will be one of the solutions by the Dijkstra's method (since there are many equi-distance solutions), however we all know that the true solution will be the red arrow line.

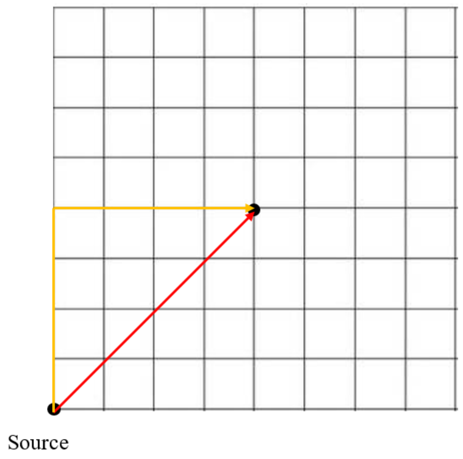


Figure 4.4 Dijkstra's failure

In conclusion, we want to use the idea of Dijkstra's algorithm to create fast method, however, we need to preserve the properties of continuous problems. The following Fast Marching Method satisfies these requirements.

4.3 Fast Marching Method

The Fast Marching method was brought up by J. A. Sethian [17], first designed for the level set method for front propagation problem.

The front propagation problem considers a boundary, usually a curve in $\mathbb{R}^2$ or a surface in $\mathbb{R}^3$. As time goes by, this boundary will evolve and move, usually back or forth along its normal directions [17]. The level set method considers the initial curve or surface as the zero level set of some function and then solve a different equation. If we consider the time function $u(\mathbf{x})$ as the time when the front reach the point $\mathbf{x}$, then it will be equivalent to the Eikonal equation.

The former section has discussed about the casusality in the Dijkstra's method. For a two dimensional method, each point has four neighbors and six for three dimensions.

The causality tells us that not all the neighbors are needed to compute the value on a specific point. For example, in two dimensional problems, there are eight fundamental directions of the characteristics: the north, south, west east and the northeast, northwest, southeast and southwest. If at a point x_i the characteristic points towards the northeast, then it means that the value function's gradient points towards this direction and due to the causality we understand that only the west and south neighbors are needed. They actually have smaller value functions than x_i.

There are actually two equivalent ways of constructing the numerical theme of Fast Marching Method. One of them is by Tsitsiklis, which uses a Semi-Lagrangian technique at each point and a first order approximation to search for the optimal point [23]. The other is by the following upwinding finite difference discretization.

$$\max\{D_{i,j}^{-x}U, -D_{i,j}^{+x}U, 0\}^2 + \max\{D_{i,j}^{-y}U, -D_{i,j}^{+y}U, 0\}^2 = F_{i,j}^{-2} \tag{4-2}$$

$$D_{i,j}^{-x}U = \frac{U_{i,j} - U_{i-1,j}}{h}, \quad D_{i,j}^{+x}U = \frac{U_{i+1,j} - U_{i,j}}{h}$$

$$D_{i,j}^{-y}U = \frac{U_{i,j} - U_{i,j-1}}{h}, \quad D_{i,j}^{+y}U = \frac{U_{i,j+1} - U_{i,j}}{h}$$

This scheme is first-order, first proposed in the iteration method of Eikonal equation and based on a Godunov-type monotone upwinding scheme. There are many analogues between the hyperbolic conservation law and the optimal control theory. The convergence result of this scheme can be seen in [1].

To solve equation 4-2 at each point, we need to assume that both terms do not vanish and solve a quadratic equation first. Then check if the causality is satisfied, that is, if the two terms really do not vanish. If not, then we need to switch to a "one-sided" updating. This happens when the characteristics are along the axes. At each step of updating ,we all need to make sure that the causality is satisfied.

The whole structure of the Fast Marching Method is now established. We can use the following table to summarize its process. [19]

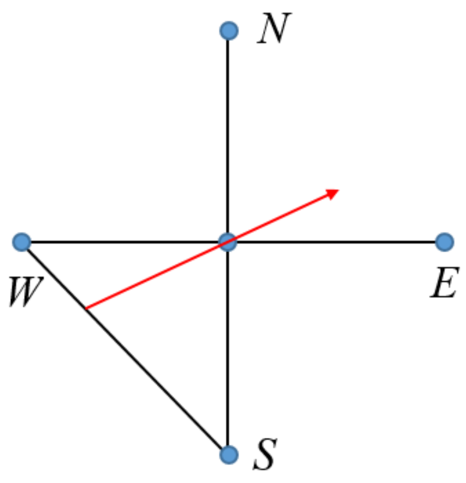


Figure 4.5 Quadratic equation

Fast Marching Method

Input: initial set Q, speed function $f(\mathbf{x})$

1. Add all the neighbors of Q into the minimum heap;
2. Check if the candidate list is empty;
3. If not, then extract the smallest $\mathbf{x_i}$ in heap;
4. For all the neighbors $\tilde{\mathbf{x}}$ of $\mathbf{x_i}$, solve the qudradtic equation for $u(\tilde{\mathbf{x}})$;
5. Check the causality: If satisfied, update $u(\tilde{\mathbf{x}})$; Else, use a one-side rule to update $u(\tilde{\mathbf{x}})$;
6. Go back to step 2.

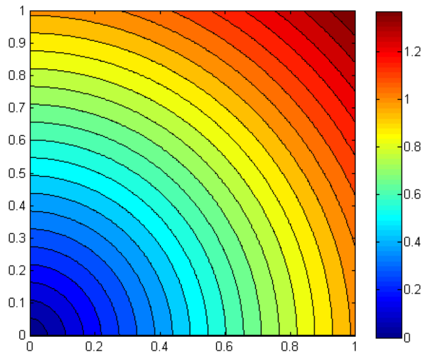
4.4 Numerical Examples

In this section we will provide several numerical examples of the Eikonal equation, with different speed functions and initial conditions.

4.4.1 Single Point Source

The following two examples have only one single point as an initial source. One of their speed function is just constantly one and the other is an oscillating sin-sin function.

$$\begin{cases} \|\nabla u(\mathbf{x})\| f(\mathbf{x}) = 1, & \mathbf{x} \in \Omega \\ u(\mathbf{x}) = 0, & \mathbf{x} = (0,0) \end{cases} \tag{4-3}$$


(a) $f(x) = 1$

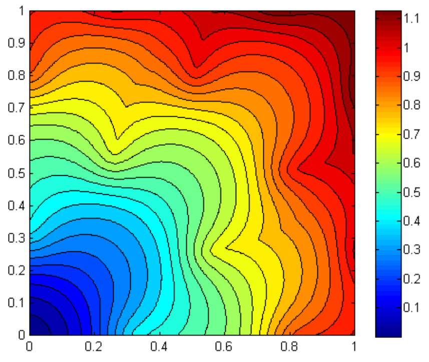
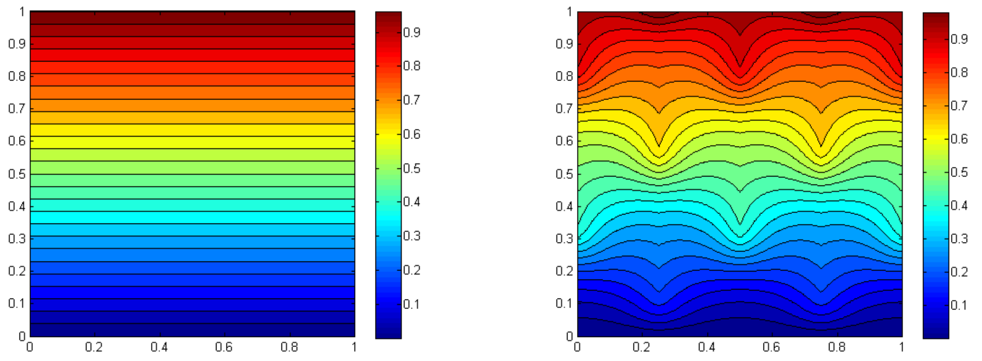

(b) $f(x) = 1 + 0.5\sin(4\pi x)\sin(4\pi y)$

Figure 4.6 Fast marching method

4.4.2 One Side Source

The following two examples have initial condition on the x-axis. One of their speed function is just constantly one and the other is an oscillating sin-sin function.

$$\begin{cases} \|\nabla u(\mathbf{x})\| f(\mathbf{x}) = 1, & \mathbf{x} \in \Omega \\ u(\mathbf{x}) = 0, & \mathbf{x} \in [0,1] \times \{y = 0\} \end{cases} \tag{4-4}$$



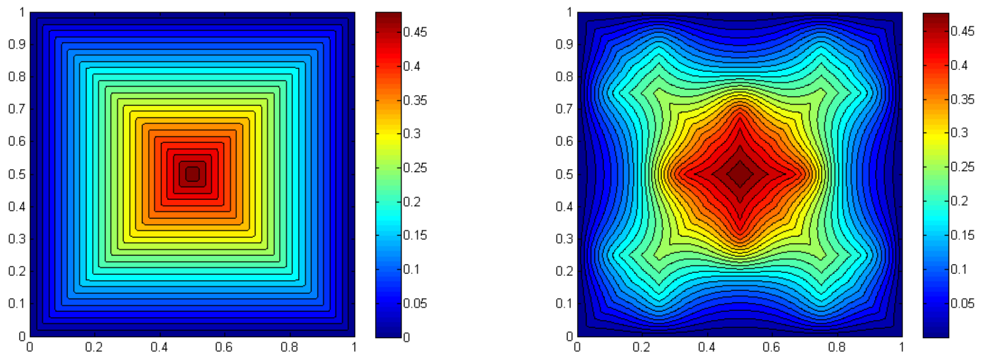
(a) $f(x) = 1$ (b) $f(x) = 1 + 0.5\sin(4\pi x)\sin(4\pi y)$

Figure 4.7 Fast marching method

4.4.3 Boundary Source

The following two examples have initial condition on the whole boundary. One of there speed function is just constantly one and the other is an oscillating sin-sin function.

$$\begin{cases} \|\nabla u(\mathbf{x})\| f(\mathbf{x}) = 1, & \mathbf{x} \in \Omega \\ u(\mathbf{x}) = 0, & \mathbf{x} \in \partial\{[0,1] \times [0,1]\} \end{cases} \tag{4-5}$$



(a) $f(x) = 1$ (b) $f(x) = 1 + 0.5\sin(4\pi x)\sin(4\pi y)$

Figure 4.8 Fast marching method

4.5 Building the Optimal Trajectories

In this section, we will assume that for each problem, we have already solved the problem and obtain the value function $u(\mathbf{x})$. We want to use this value function to build the optimal trajectory for each point. The main tool to implement is still the Bellman equation:

$$\begin{cases} u(\mathbf{x}) = \inf\limits_{\mathbf{a}(\cdot)\in\mathcal{A}} \left\{\tau + u\left(\mathbf{y}(\tau)\right)\right\}, & \text{if } \mathbf{x} \in \Omega \\ u(\mathbf{x}) = q(\mathbf{x}), & \text{if } \mathbf{x} \in \partial\Omega \end{cases} \tag{4-6}$$

Notice that before we solve the equaiton, the value function is unknown and we cannot use the Bellman equation directly. However, now since we already have the value function, we can just define a small value τ as the radius of a small circle, and on each circle we find the minization point of the value $\tau + u\left(\mathbf{y}(\tau)\right)$. Then we can draw a line between the present point and this point and this is a small piece of the optimal trajectory. We can repeat this process for each point until we reach the target set (or initial conditions).

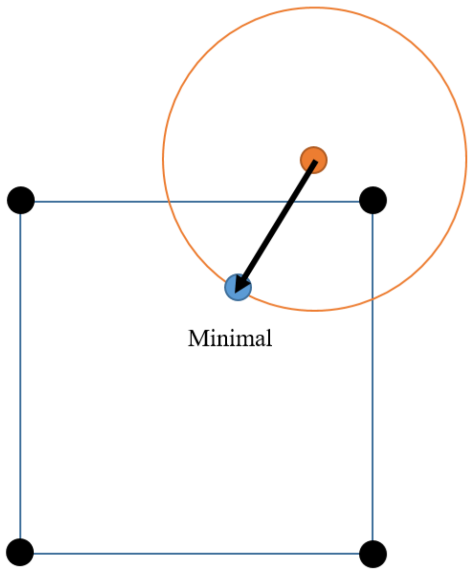


Figure 4.9 Search on circle

Notice that we actually cannot find the exact solution of the minimization problem since we do not have all the values on the circle. We only have the value functions on each grid point. Therefore, to obtain the value function for some points on the circle, we need to make an interpolation. Usually the first order bilinear interpolation (using the four grid points of which square unit the point belongs to) will be necessary. Besides, we cannot obtain the value function on all the points of the circle, we need to choose a number M of points and do the minimization among these points.

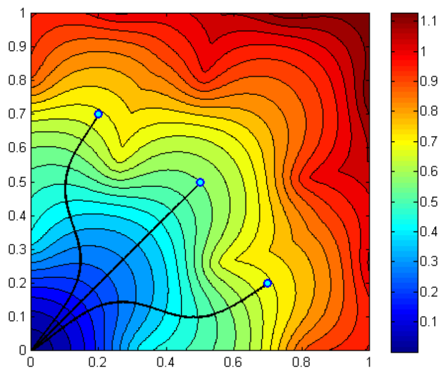


Figure 4.10 Optimal trajectories

An alternative to the former method is to use the Golden-Section Search on the circle, which is very efficient.

The golden-section search method is an advanced minimization method of the classical bisection method. It can help find the extremum of a strictly unimodal function. Instead of a bisection process, the golden-section search intends to use a "tri-section", which leads to the use of golden ratio. In fact, using a golden ratio is a necessary condition of trying to reducing the points we want to use to cut the interval. Each time we want to cut the interval into three pieces, which means that we will need two new points. However, if we use the golden ratio to cut the interval, one of the points used in the former section will still be useful. This is the main idea of Golden-Section Search algorithm.

The following is a brief introduction of the golden-section search algorithm.

Golden-Section Search (Minimization)

Input: function $f(x)$, interval $[a, b]$, golden ratio $\alpha = \frac{1+\sqrt{5}}{2}$

1. Let $c = b - \frac{b-a}{\alpha}$, $d = a + \frac{b-a}{\alpha}$, and compute $f(c)$ and $f(d)$;
2. If $f(c) < f(d)$, then replace b as d and d as c. Update $c = b - \frac{b-a}{\alpha}$;
3. Else, replace a as c and c as d. Update $d = a + \frac{b-a}{\alpha}$;
4. Continue this process until the error is small enough.

4.6 Conclusion

In this chapter, we start with the iterative method of the numerical solution of Eikonal equation. The iterative method is not what we really want. Then we change the topic to the discrete version of shortest path problem, discussing different algorithms used to solve the shortest path problem on graph: the generic shortest path algorithm, the Bellman-Ford algorithm and the Dijkstra's method. The idea of Dijkstra's algorithm is quite important, even if it cannot be directly used to solve a continous problem. The Fast Marching Method is an efficient method of solving the Eikonal equation, which is a Dijkstra-like method, based on a first order Godunov upwinding scheme. We also provided several numerical examples based on different initial conditions. At last, we show how to draw the optimal trajectory of each point. The main idea is to use the Bellman equation.

Chapter 5 Rarefaction Fan in Eikonal Equation

In this chapter, we will mainly talk about the rarefaction fan. Firstly, we introduce the rarefaction fan in the hyperbolic conservation laws, especially in Riemann problem. Then we will compare it with the rarefaction fan which appears in solving the Eikonal equation. We will explain why the rarefaction fan is terrible for the original Fast Marching Method.

5.1 Riemann Problem

The rarefaction fan arises from the hyperbolic conservation law, where it is usually called the rarefaction wave, especially the Riemann's Problem [21]. In fact there are two common phenomena in the Riemann problem: the shock wave and the rarefaction wave. The shock wave considers the problem when two characteristics hit each other while the rarefaction wave considers when two characteristics head away from each other. These two phenomena happen when there is a jump in the initial data.

When considering the system of conservation laws [8]

$$\mathbf{u}_t + \mathbf{F}(\mathbf{u})_x = 0, \quad \text{in } \mathbb{R} \times (0, \infty) \tag{5-1}$$

If the initial condition is piecewise-constant, i.e.

$$\mathbf{g} = \begin{cases} u_l, & \text{if } x < 0 \\ u_r, & \text{if } x > 0 \end{cases}$$

Suppose that the initial condition satisfies $u_l < u_r$, then use the method of lines of characteristics, we see that the solution looks like 5.1

Around the origin, where the initial data is discontinuous, we see a series of characteristic lines emitting from the same point. In fact there is a sector region where the characteristic lines are not defined, while to assure that the solution is continuous, we need to continuously combine the two parts of the solution together. This "fan" is

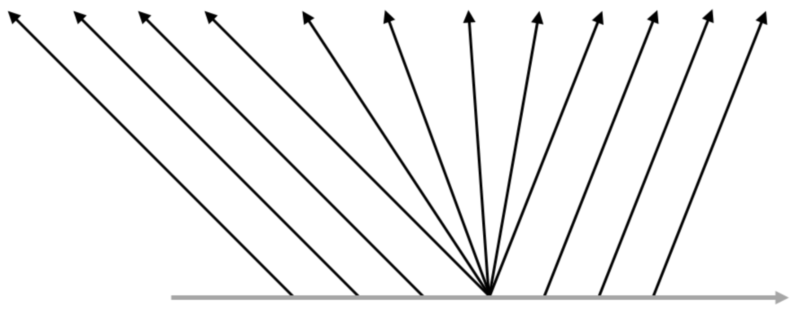


Figure 5.1 Rarefaction wave

actually created by this continuation.

Whenever there is a bunch of characteristics emitting from the same point, then we will call that there happens a rarefaction fan.

The counterpart of the rarefaction wave is the shock wave, where the characteristics collide with each other. The initial condition now needs to satisfy $u_l > u_r$. The shock wave looks something like the following figure:

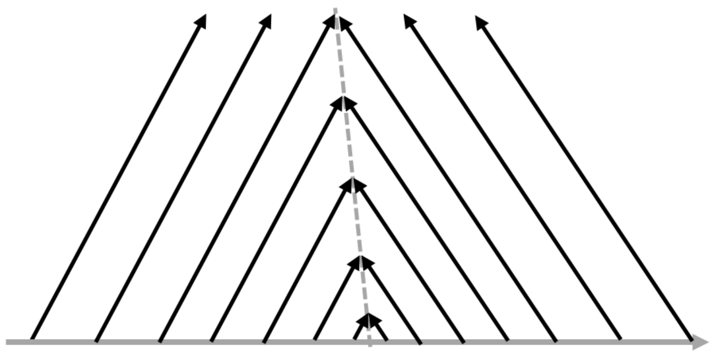


Figure 5.2 Shock wave

Later we will see that for the solution of Eikonal equation, there also appears some collisions of characteristics which are very similar to the shock wave. However, the original Fast Marching Methdo does not have any problems around the collisions of characteristics. It is the rarefaction fan that brings trouble and will reduce the convergence rate.

5.2 Eikonal Equation: Single Source

Similar things happen when we are solving the Eikonal equation. If we consider an Eikonal equation with an isolated single point as the initial condition, we can also observe a rarefaction fan around this initial point. Consider

$$\begin{cases} \|\nabla u(\mathbf{x})\| f(\mathbf{x}) = 1, & \mathbf{x} \in \Omega \\ u(\mathbf{x}) = 0, & \mathbf{x} \in \Omega_0 \end{cases} \tag{5-2}$$

If $f(\mathbf{x}) = 1$, and Ω_0 contains only a single point, then the value function $u(\mathbf{x})$ gives us the distance towards that target point.

The rarefaction create a cone around the source. Assume $f(\mathbf{x}) = 1$, the true solution is

$$u(x, y) = \sqrt{x^2 + y^2}$$

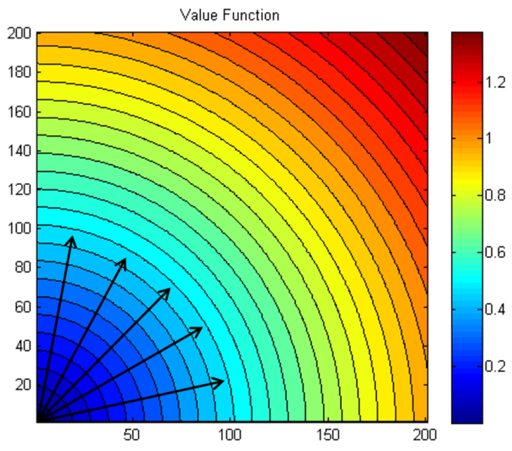


Figure 5.3 Rarefaction fan

If we find the second-order partial derivatives, they will be

$$\frac{\partial^2 u}{\partial x^2} = \frac{y^2}{(x^2 + y^2)^{\frac{3}{2}}}$$

$$\frac{\partial^2 u}{\partial y^2} = \frac{x^2}{(x^2 + y^2)^{\frac{3}{2}}}$$

If $x, y \neq 0$, they will tend to infinity as $x^2 + y^2 \to 0$. Since the numerical scheme is based on a first order approximation, the blow up of the second-order derivatives will result in some pollution errors and it cannot achieve first order convergence.

If we use the original version of Fast Marching Method to solve this problem and we make a convergence study, then we will not obtain the first order convergence shown in the following picture.

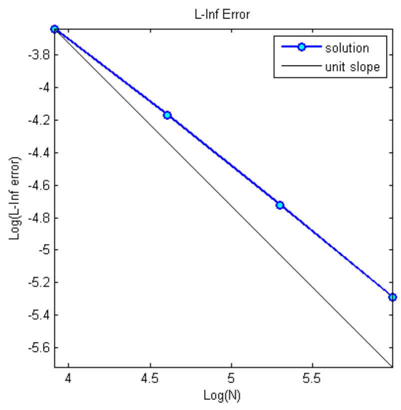


Figure 5.4 Convergence study

In the above convergence study we can see that the line of convergence is deviated from the line of slope one, therefore the first order convergence is failed.

5.3 Conclusion

In this chapter, we start with two very common phenomena in the hyperbolic conservation law: the shock wave and the rarefaction wave. The rarefaction wave is quite similar to the rarefaction fan which appears in solving the Eikonal equation. Roughly speaking, the rarefaction fan means there are many characteristics emitting from the same point. This phenomenon leads to the second order derivatives tending to infinity around the source, which is annoying in the computation. The next chapters will try to fix this problem.

Chapter 6 Lagrangian Technique and Factored Eikonal Equation

Due to the appearence of the rarefaction fan, the original Fast Marching method cannot have first order convergence. In this section, we will show two ways of dealing with the rarefaction fan caused by the single source, the Lagrangian technique and the factored Eikonal equation. A numerical example and a convergence study will be provided.

6.1 Lagrangian Technique

In the case where the speed function is constant, we already know that the characteristics are straight lines. A quite naive idea is that we can integrate along the characteristics to get the exact value on each point. It seems like that we are "shooting" the characteristics towards some points. We can do this technique in a small neighborhood with a fixed radius of the source and after that we can switch back to the original Fast Marching Method.

However, if the speed function is not constant, an integral along a straight line will not provide the exact solution. This is because that the characteristics are no longer straight lines, and pretending they are straight lines will bring in larger errors especially when the characteristics change directions quite often. There might be another alternative of attempting to use the "shooting method" at the rarefaction fan, that is to use the two point boundary value problem solvers to try to solve the ordinary differential equations (or by Pontryagin Maximum Principle) to obtain the value on those points. But to solve such a problem at so many points will be a very time-consuming task.

We will give an example in the next section where we can see that as the grid size become smaller, the error caused by using straight lines will dominate and the method is not able to recover first order convergence.

Remark 6.1. *We call this technique the "Lagrangian Technique" due to the "Lagrangian specification of the field" in the theory of fluid dynamics [11].*

6.2 Factored Eikonal Equation

In this section we will introduce the factored Eikonal equation and how it works. Consider a two dimensional Eikonal equation with the initial source containing only one point

$$\begin{cases} \|\nabla u(\mathbf{x})\| f(\mathbf{x}) = 1, & \mathbf{x} \in \Omega \\ u(\mathbf{x}) = 0, & \mathbf{x} \in \mathbf{x_0} \end{cases} \tag{6-1}$$

Now since we have already had a sence on how the solution look like asymptotically around the source, we might use some factor functions to help cancel out the singularities. The idea is to factor the original value function into two functions. One of them is a known function containing the singularity while the other will be smooth. If we solve the equation with respect to the new function, then there will not be "blow up" of the second derivatives to the smooth function and this might bring back the first order convergence.

There have been already some tests on the factored Eikonal equaitons. According to [13], there are two ways to factor, for example the multiplicative factor,

$$u(\mathbf{x}) = \tau_0(\mathbf{x})\tau_1(\mathbf{x})$$

and the addtional factor,

$$u(\mathbf{x}) = \tau_0(\mathbf{x}) + \tau_1(\mathbf{x})$$

where $\tau_0 = \frac{\|\mathbf{x}-\mathbf{x_0}\|}{f(\mathbf{x_0})}$. This is how the solution looks like asymptotically to the source. We choose this τ_0 because this is the true solution when the speed function $f(\mathbf{x})$ is constant. If it is not, locally it can still approximate that function. This is a rather local problem since we can see that how the rarefaction fan looks like only depends on the value of the speed function on the source $\mathbf{x_0}$. Then the original PDE becomes the "factored Eikonal equation"

$$\|\nabla\tau_0(\mathbf{x})\tau_1(\mathbf{x}) + \nabla\tau_1(\mathbf{x})\tau_0(\mathbf{x})\| f(\mathbf{x}) = 1 \tag{6-2}$$

and

$$\|\nabla\tau_0(\mathbf{x}) + \nabla\tau_1(\mathbf{x})\| f(\mathbf{x}) = 1 \tag{6-3}$$

6.2.1 Modified Fast Marching Method

The numerical scheme for 6-2 and 6-3 are similar to the Fast Marching Method, however, we are now solving quadratic equations for τ_1, not for $\tau_0 + \tau_1$. This means we need a slightly different change in the numerical scheme. The original scheme can be written as [22]

$$\max\{D_{i,j}^{-x}u, -D_{i,j}^{+x}u, 0\}^2 + \max\{D_{i,j}^{-y}u, -D_{i,j}^{+y}u, 0\}^2 = \frac{1}{f_{i,j}}$$

According to equation 6-2 and 6-3, the numerical operator now acts on τ_1 and contains some terms of τ_0. We will list down the new numerical operators. For example, in the multiplicative case, the two operators for x variable are

$$D_{i,j}^{-x}\tau_1 = (\nabla\tau_0)_{i,j}(\tau_1)_{i,j} + \frac{(\tau_1)_{i,j} - (\tau_1)_{i-1,j}}{h}(\tau_0)_{i,j}$$
$$D_{i,j}^{+x}\tau_1 = (\nabla\tau_0)_{i,j}(\tau_1)_{i,j} + \frac{(\tau_1)_{i+1,j} - (\tau_1)_{i,j}}{h}(\tau_0)_{i,j}$$

As for the additional case, the two operators for x variable are

$$D_{i,j}^{-x}\tau_1 = (\nabla\tau_0)_{i,j} + \frac{(\tau_1)_{i,j} - (\tau_1)_{i-1,j}}{h}$$

$$D_{i,j}^{+x}\tau_1 = (\nabla\tau_0)_{i,j} + \frac{(\tau_1)_{i+1,j} - (\tau_1)_{i,j}}{h}$$

We should pay attention to which value the minimum heap is using to sort the candidate set. Since we use the value function to sort the points in candidate set in the original verstion of Fast Marching Method, we should still use it here. That is, to use $\tau_0\tau_1$ or $\tau_0 + \tau_1$ rather than τ_1 to sort the minimum heap. The value function is the reason for causality, so that is why we need to use $\tau_0\tau_1$ or $\tau_0 + \tau_1$ rather than τ_1.

Former discussions gives rise to the following "modified Fast Marching Method" algorithm [22] [13]. Here we use the "additional factor" version of τ_0. For the multiplicative version of factored Eikonal equation, it is quite similar.

Modified Fast Marching Method

```
Input: source point $\mathbf{x_0}$, speed function $f(\mathbf{x})$, factor function $\tau_0(\mathbf{x})$
1. Add $\mathbf{x_0}$ into the minimum heap;
2. Check if the candidate list is empty;
3. If not, then extract the smallest $\mathbf{x_i}$ in heap based on $\tau_0 + \tau_1$;
4. For all the neighbors $\tilde{\mathbf{x}}$ of $\mathbf{x_i}$, solve the qudradtic equation for $\tau_1(\tilde{\mathbf{x}})$;
5. Check the causality: If satisfied, update $u(\tilde{\mathbf{x}})$ as $\tau_0(\tilde{\mathbf{x}})+\tau_1(\tilde{\mathbf{x}})$; Else, use a one-side
   rule to compute $\tau_1(\tilde{\mathbf{x}})$, and update $u(\tilde{\mathbf{x}})$ as $\tau_0(\tilde{\mathbf{x}}) + \tau_1(\tilde{\mathbf{x}})$;
6. Go back to step 2.
```

As we can see in this algorithm, since we are using $\tau_0 + \tau_1$ to sort the candidate set, the shape of τ_0 does not influence the propagation of our candidate set as long as it can help reduce the error.

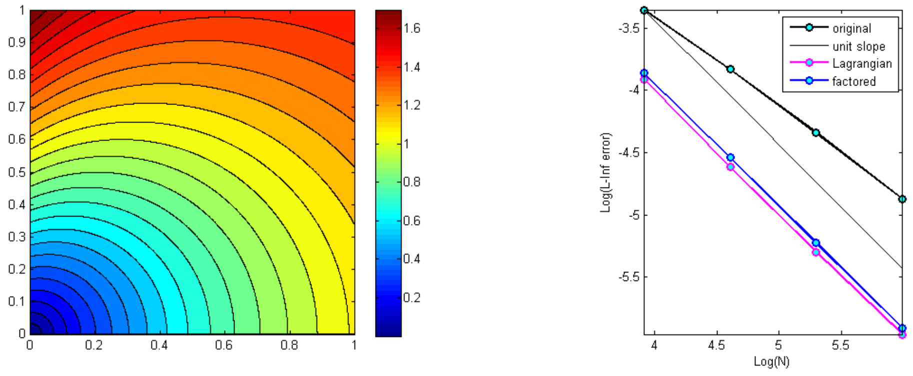
6.3 Numerical Example

One thing we should notice is that since we choose the τ_0 as the distance function, we will get the exact solution if the speed function is constant. Therefore we cannot obtain a convergence study in this case (The errors will be within accuracy of machine computation). Here is an example with an inhomogeneous speed function. We can obtain the analytic solution.

$$f(\mathbf{x}) = \frac{1}{s_0} + a\mathbf{e} \cdot (\mathbf{x} - \mathbf{x_0})$$

$$u(\mathbf{x}) = \frac{1}{a}\text{arccosh}\left(1 + \frac{1}{2}s_0 a^2 \frac{\|\mathbf{x} - \mathbf{x_0}\|^2}{f(\mathbf{x})}\right)$$

Choose $s_0 = 2, a = 1, x_0 = (0,0), \mathbf{e} = (1,0)$. 6.1 contains a convergence study of the two methods



(a) level set (b) L^∞ error

Figure 6.1 Lagrangian & factored Eikonal equation

Here the thin black line has slope -1 and is for comparison. We can see that both method is able to improve the convergence rate and reduce the errors. There is no obvious advantages of either of the techniques here in this problem. However, in the next section, we will see the difference between this two methods.

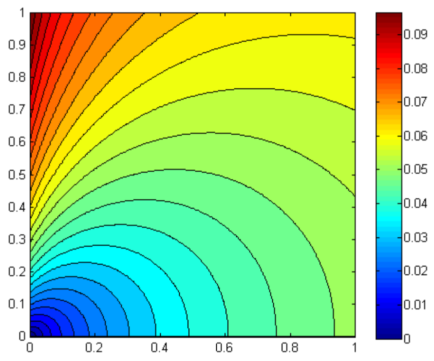
6.4 Lagrangian Technique vs. factored Eikonal equation

In this section, let us make a comparison between the Lagrangian technique and the factored Eikonal equation, and analyze their behavior in a slightly different problem.

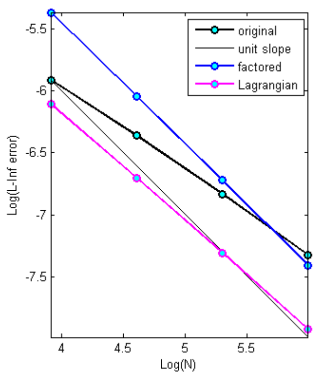
Consider the former problem with different parameter $a = 50, s = 0.25$. If we plot down the level set of the value function, we will see that at the region far away from the source, the characteristics will be quite different from the factor function τ_0. As a result, it is unnatural to use a distance function here to factor the value function. Although this does not influence the convergence rate, using a factored Eikonal equation in the whole domain will bring large errors.

As for the Lagrangian technique, the characteristics are no longer straight lines now, and they are not known a priori. (even if in the former example the characteristics are not straight lines but in this problem the characteristics deviate from the straight lines more) If we insist to use a straight line as the characteristic, then we cannot get the true value on each point this time. Therefore the convergence rate for this technique will become worse. Remeber that in the shortest path problems, we always update the value on each point using the smallest possible value, that is, each time we update a value, we compare the current value on this point to the new one and try to accept the smaller one. Therefore, this process can slightly help reduce the error caused by the Lagrangian technique since the Lagrangian technique will get larger values, but when we use the original method to update, it is possible to reduce the value and the former value will be replaced.

The following is the level set of value function and the visualization of convergence study. We can see that although the Lagrangian technique still helps reduce the error in a large scale, it loses the first order convergence. The convergence rate of Lagrangian technique is smaller than the factored Eikonal equation. However, even if the factored Eikonal equation preserves the first order convergence, it might bring large errors. This is due to the fact that far away from the source, the characteristics are quite different from the gradients of τ_0 and insisting on using the factoring technique here seems to be not very reasonable.



(a) level set



(b) L^∞ error

Figure 6.2 Lagrangian vs. factored Eikonal equation

Since the Lagrangian technique loses the first order convergence, in the following algorithm we will no longer use it.

6.5 Conclusion

In this chapter we start with the “Lagrangian technique”, which is the first try to resolve the problem of rarefaction fan. It tries to “shoot” some characteristic around the source and integrate to obtain a possible value. Then we introduce the factored Eikonal equation. This method is trying to factor the original value function into two parts and let one of them containing the singularity while the other will be smooth. We show how this new equation can be solved and provide the algorithm with respect to it. A numerical example is also provided. In the next part we explain why the Lagrangian technique will fail in some situations and we give an example where the factored Eikonal equation can still obtain first order convergence while the Lagrangian technique cannot.

Chapter 7 Rarefaction Fan Caused by Obstacles

As we have discussed before, there are many other reasons that might cause a rarefaction fan to happen. Since whenever there is a series of characteristics coming from the same source, there will be a rarefaction fan, an easy guess will be the rectangular obstacle. In this section, we will first discuss about the rarefaction fan created by the rectangular obstacles and then give the definition of “bad corners”. After that we will talk about the weird behaviors of optimal trajectories.

7.1 Rectangular Obstacles

Consider the distance function on some domain in two dimension. If there are obstacles in the domain, then the shortest paths for some points “hiding behind the obstacles” will have to pass through the corners of that obstacles. Then there will be a bunch of characteristics emitting from these corners.

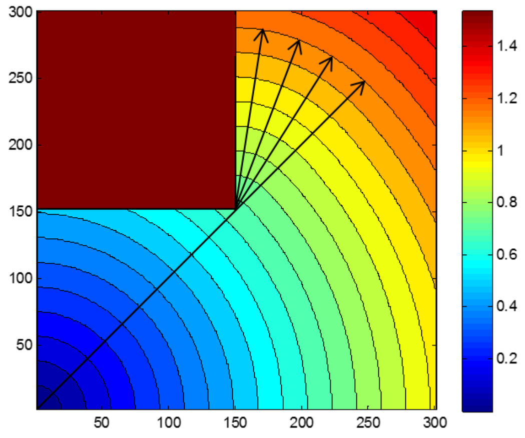


Figure 7.1 Rarefaction fan

However, not every corner can cause the rarefaction fan. The important condition of causing a rarefaction fan at a corner is that the characteristic “passes by” the corner, rather than head towards the obstacle. This actually gives us the condition of how a corner can cause the rarefaction fan. We may name them the “bad corners”. (In this article we only consider rectangular obstacles)

Definition 7.1. Bad corner: *A "bad corner" is a corner where the characteristic of the value function at this corner points from one non-obstacle region to another non-obstacle region, i.e. "passes by " the obstacle rather than "heads into" the obstacle.*

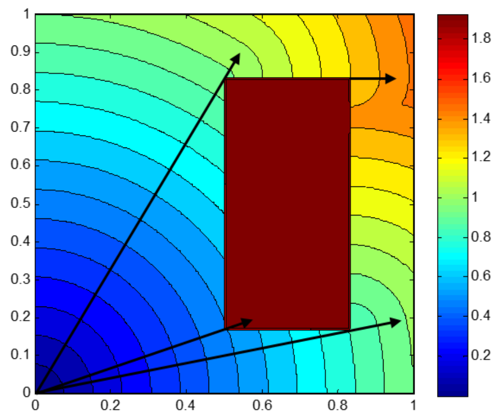


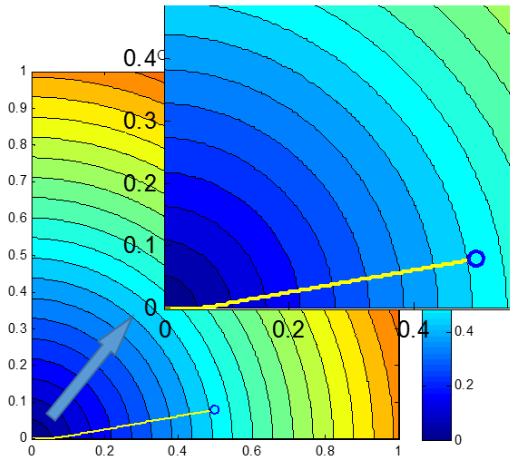
Figure 7.2 Bad corners

For example in 7.2, the lower left corner is not a "bad corner" since the characteristic line runs into the obstacle while the other threes are since the characteristic lines pass by the corners.

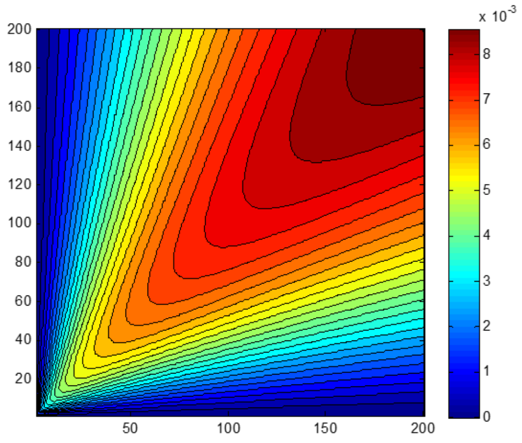
7.2 Weird Trajectories

One of the nuisances of the rarefaction fan is that the optimal trajectories, i.e. the lines of characteristics will behave weirdly. Consider the simple case where the value function is the distance function. 7.3 shows us what will be observed if we use the original Fast Marching Method to solve the problem.

Although the second derivatives of the distance function will blow up near the source, we need to notice that along the coordinate axes the derivatives will never blow up and they are always zero. Therefore we can obtain quite exact solutions along the axes. On the contrary, the numerical solution will have quite large errors inside the quadrant. 7.3(b) shows us the error distribution on the whole domain. As a result, if we use Bellman's optimal principle to find the optimal trajectory of a point, it will not go straight to the source. Instead, they will first hit the coordinate axes and then go along the axes until they reach the source.



(a) Trajectory



(b) Error distribution

Figure 7.3 Weird trajectory

Similar things happen around the “bad corners”. The optimal trajectories will hit the boundary of the obstacles first and then go along the boundary until they reach the corner. 7.4 and 7.5 are the illustration of this phenomenon.

In 7.5 the “maze navigation” problem, there are many “bad corners” and we can see at each “bad corners” the trajectories behave weirdly.

One thing we need to notice is that since there is still a rarefaction fan around the source, we see that the trajectories will also behave weirdly around the source.

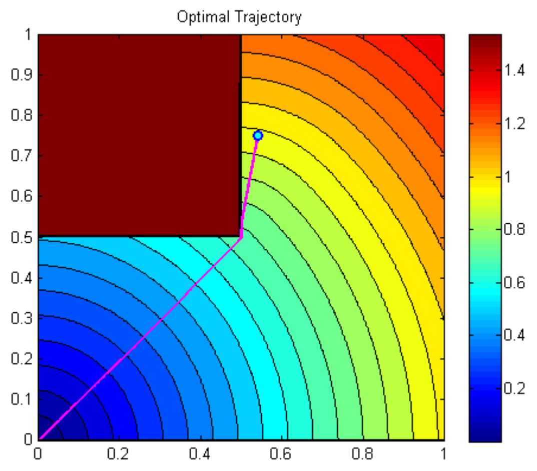


Figure 7.4 Weird trajectory

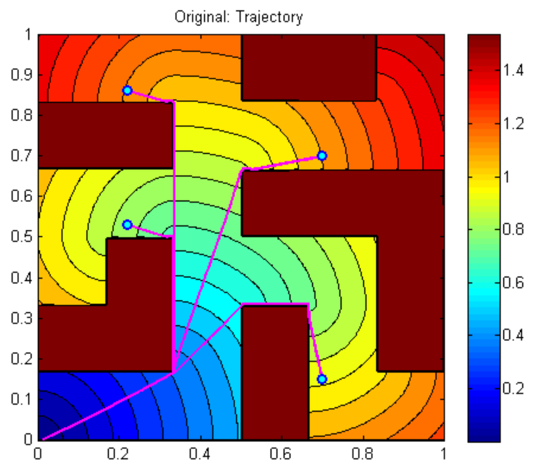


Figure 7.5 Weird trajectory

7.3 Conclusion

In this chapter, we first discuss about the rarefaction fan caused by the rectangular obstacles and then give the definition of “bad corners”. Then we show the weird behaviors of the optimal trajectories, caused by the loss of accuracy of the original Fast Marching Method. This phenomenon is more obvious if we are solving the maze navigation problem.

Chapter 8 Algorithm and Numerical Examples

According to all the knowledge in former chapters, in this section, we are goint to propose the higher accuracy path planning algorithm, especially appropriate to solve the problem with obstacles. We are going to provide some numerical examples.

8.1 Algorithm

Suppose we are solving the Eikonal equation possiblely with several rectangular obstacles (we call them the set of obstacles $\mathcal{B}$) appearing in the domain. Now we want to use the method of factored Eikonal equation each time we hit a "bad corner". Because there might be several "bad corners", we also need to determine to which point we want to build the τ_0 and use the factored Eikonal equation.

$$f(\mathbf{x}) = \begin{cases} 1, & \mathbf{x} \notin \mathcal{B} \\ +\infty, & \mathbf{x} \in \mathcal{B} \end{cases}$$

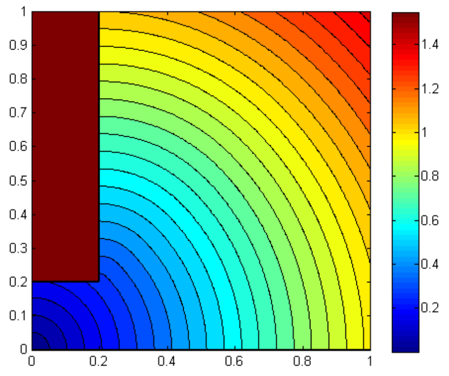
Higher Accuracy Path Planning Algorithm

```
Input: source point $\mathbf{x_0}$, speed function $f(\mathbf{x})$, several rectangular obstacles
1. Add $\mathbf{x_0}$ into the minimum heap;
2. Check if the candidate list is empty;
3. If not, then extract the smallest $\mathbf{x_i}$ in heap based on $\tau_0 + \tau_1$;
4. For all the neighbors $\tilde{\mathbf{x}}$ of $\mathbf{x_i}$
    • If $\tilde{\mathbf{x}}$ is around a "bad corner", use factored Eikonal equation to update
    • Else, use the original FMM to update
5. Go back to step 2.
```

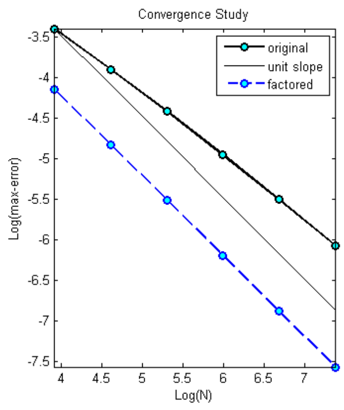
8.2 Single Obstacle

The first example is a Eikona equation with the speed function $f(\mathbf{x}) = 1$ and one single obstacle appearing in the domain. The exact solution is only the distance function toward the origin in this case. Due to the appearence of the obstacle, there will be a rarefaction fan within a large region: $\{y > x, x > 0.2\}$.

This example is computed on grids with $50 \times 50, 100 \times 100, 200 \times 200, 400 \times 400, 800 \times 800, 1600 \times 1600$ points each. For the convergence study, we use the L^∞ error and plot down the log-log picture. We will also draw a line with slope of -1 to help compare the convergence rate of different method. It will be clear if the method helps recover the first order convergence.



(a) level set

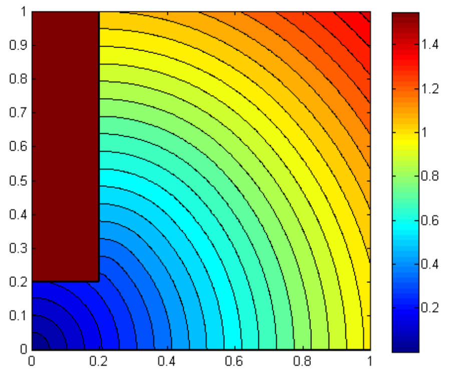


(b) convergence

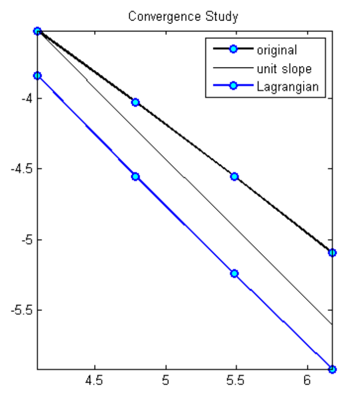
Figure 8.1 Single obstacle

We can see from 8.1 that the algorithm performs very well. It can help recover the first order convergence and help reduce the error. One thing we need to notice is that there are actually two different rarefaction fans here: one of them is located at the origin and the other around the bad corner. We want to deal with them separately by factored Eikonal equation.

The Lagrangian technique can actually also be applied to this problem, since we are solving the equation with $f(\mathbf{x}) = 1$. The following is a computing result of the same example above, using the Lagrangian technique. A convergence study is also provided here since the exact solution is quite easy to find. This example is computed on grids with $50 \times 50, 100 \times 100, 200 \times 200, 400 \times 400$ points each. For the convergence study, we use the L^∞ error.



(a) level set



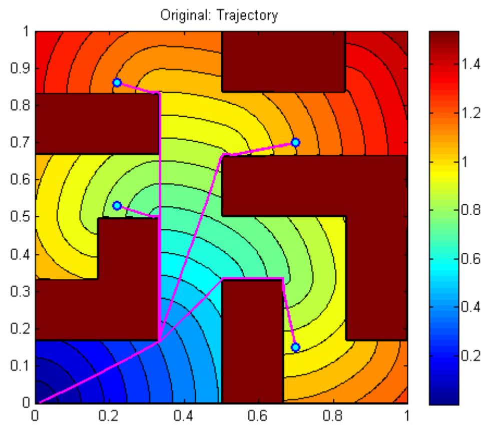
(b) convergence

Figure 8.2 Single obstacle

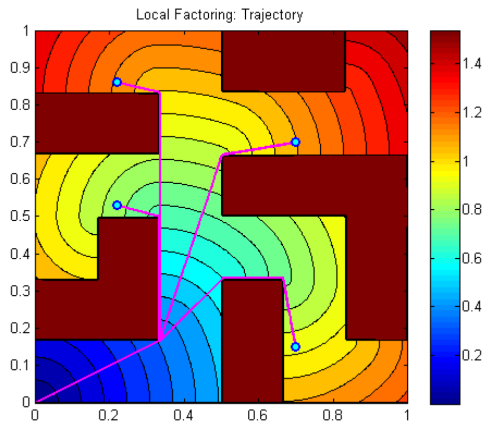
8.3 Maze Navigation

In this section, we will apply the former algorithm to the maze navigation problem and show the improvement of the weird behavior of trajectories.

The maze navigation problem is actually solving the Eikonal equation, with many rectangular or combinations of rectangular obstacles appearing in the domain. Here we still consider a continuous case of path planning. When the maze is quite complicated, it is almost impossible to find the analytic solution. Therefore, it is better to show the improvement of trajectories.



(a) original



(b) new

Figure 8.3 Improvement of trajectories

In the above two figures, the upper one uses an original Fast Marching Method, where we can observe the weird behaviors of the trajectories. The lower one uses the algorithm we proposed in 8.1, where we can see that the trajectories go directly to the corners instead of running towards the boundary first. This improvement proves that around the rarefaction fans the errors are reduced and it means that our algorithm is of higher accuracy.

Since we are solving a distance function, the Lagrangian technique can still be used to help improve the convergence rate. According to our former discussion, if we have an inhomogeneous speed function here and we still want to solve a maze navigation problem, then straight lines will bring errors and we might not obtain higher accuracy.

The following is the same maze navigation problem shown above. We use the Lagrangian technique at each point and draw many optimal trajectories. It is easy to observe that there is also a big improvement in the trajectories.

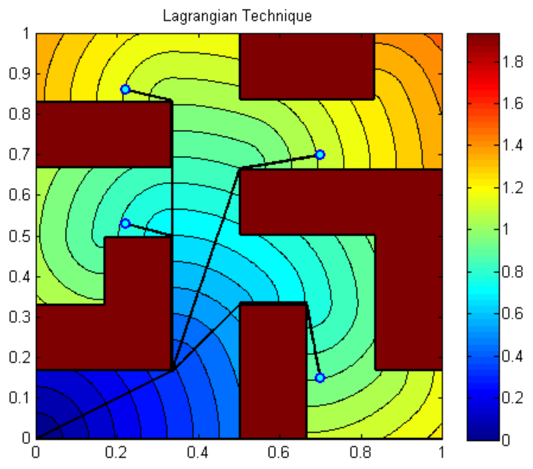


Figure 8.4 Lagrangian technique

8.4 Conclusion

In the chapter, we start by giving the algorithm for solving the path planning problem with several obstacles appearing in the domain. The algorithm simply uses the factored Eikonal equation to deal with the rarefaction fan each time we hit a “bad corner”. Since we are dealing with the distance function case, where $f(\mathbf{x}) = 1$ so the “Lagrangian technique” still works here. We provide two numerical examples with only one obstacle appearing in the domain and there will be two rarefaction fans needed to be dealt with: the one around the source and the one around the only bad corner. The convergence study shows that both methods can help recover the first order convergence. Then in the final part, we apply this algorithm to a more complicated problem “maze navigation” where there are several obstacles appearing in the domain and it looks like a maze. Since the analytic solution is difficult to find, we plot down several optimal trajectories of different points and the improvement shows that our algorithm can help reduce the errors. This means that our initial expectation is fulfilled.

Chapter 9 Conclusions

In this section, we will make some conclusions about thesis and list down some future problems to be solved.

9.1 Achievements

The path planning problem has many applications in the field such as the robotics. The continuous path planning problem is done by solving the Eikonal equation. Usually we need to solve the problem with one single source. And sometimes there appear rectangular obstacles within the domain. These are the most common reasons we need to find the numerical method for solving the rarefaction fans. So far all the methods that exist can only deal with the single source case while this idea can be easily extended to deal with the obstacles.

In this thesis, we have briefly introduced the background of optimal control theory, dynamic programming, the Bellman equation, Hamilton-Jacobi-Bellman equation. These are the general mathematical theories of Eikonal equation. Furthermore, we have also discussed the general structure of the algorithm for discrete shortest path problem and the Fast Marhcing Method, which can be viewed as a continuous counterpart. Some numerical examples for both cases are provided. Then we showed that the original methods fail to deal with the rarefaction fans.

In the final part, we propose our new algorithm and successfully implemented the factored Eikonal equation in the obstacle case. The convergence study shows that this method can help us both reduce the errors and recover the first order convergence. Besides, we provided an algorithm based on this method to solve the “maze navigation” problem which seems like the classical maze but is defined on a continuous domain. The optimal trajectories prove that the method can largely help reduce the error.

9.2 Problems

Although our algorithm performs well in the above problem, there are still some problems to be solved. The reasons causing rarefaction fans, discontinuous speed functions and a desire to generalize it to three dimension are our most interested problems. We will briefly discuss them in this section.

9.2.1 Other Reasons Causing Rarefaction Fan

Although we understand the reason of how the rarefaction fan happens, due to the single intial source or the appearacne of the rectangular obstacles. However, there might be other reasons which can cause the rarefaction fan. For example, the rarefaction wave in Riemann's problem is caused by the discontinuity of the initial data. A very simple idea is to consider whether a discontinuous initial data for the Eikonal equation will cause a rarefaction fan? Or under what condition will the initial data create some rarefaction fans? We need to be careful about this problem since the initial data cannot be totally randomly set, it also need to satisfy the properties of a solution under the viscosity sence.

9.2.2 Discontinuous Speed Function

This part is due to the following reasoning: the usual definition of an obstacle is "regions where the moving unit cannot head into". However, we can also consider these regions as "where the speed function is constantly zero". A further question then will be brought up that if the speed function within the obstacles is not zero but some other constants different from the outside regions, will there be a rarefaction fan? This actually contradicts the condition of the speed function to be Lipschitz continuous. Since the speed function becomes discontinuous, we might need to set up new theories about the existence and uniqueness of the solution, or we might need to define new solutions. This is obviously a wider question since the obstacles can be viewed as a special case for this problem.

9.2.3 Rarefaction Fan in $\mathbb{R}^3$

The last question is difficult. Since we know that the shortest path problem can be defined in higher dimensions, and so does the rarefaction fan can happen. Therefore we want to know if the algorithm can be extended to higher dimensions? Take $\mathbb{R}^3$ as an example: the path planning problem in $\mathbb{R}^3$ is understandable. However, the difficult part is in $\mathbb{R}^3$, the rarefaction fan happens not only at some corners, but some edges. That is, around some edges there will be a rarefaction fan (a series of characteristics emitting from the same point). This makes the rarefaction fan more complicated and detecting them more cumbersome. So the question of how to generalize the former method is quite difficult.

REFERENCE

[1] Guy Barles and Panagiotis E Souganidis. Convergence of approximation schemes for fully nonlinear second order equations. *Asymptotic analysis*, 4(3):271–283, 1991.

[2] R Bellmann. Dynamic programming princeton university press. *Princeton, NJ*, 1957.

[3] Dimitri P Bertsekas. Dynamic programming and optimal control, 1995.

[4] Dimitri P Bertsekas. *Network optimization: continuous and discrete models*. Citeseer, 1998.

[5] Dimitri P Bertsekas. Abstract dynamic programming. *Athena Scientific, Belmont, MA*, 2013.

[6] Thomas H.. Cormen, Charles Eric Leiserson, Ronald L Rivest, and Clifford Stein. *Introduction to algorithms*, volume 6. MIT press Cambridge, 2001.

[7] Michael G Crandall and Pierre-Louis Lions. Viscosity solutions of hamilton-jacobi equations. *Transactions of the American Mathematical Society*, 277(1):1–42, 1983.

[8] Lawrence C Evans. *Partial differential equations; 2nd ed.* Graduate Studies in Mathematics. American Mathematical Society, Providence, RI, 2010.

[9] Wendell H Fleming and Halil Mete Soner. *Controlled Markov processes and viscosity solutions*, volume 25. Springer Science & Business Media, 2006.

[10] LD Landau, Eo M Lifshitz, and RJ Donnelly. Mechanics, vol. i of course on theoretical physics. *American Journal of Physics*, 40(7):1050–1051, 1972.

[11] Lev Davidovich Landau and Evgenii Mikhailovich Lifshitz. Fluid mechanics. 1987.

[12] D. Liberzon. *Calculus of Variations and Optimal Control Theory: A Concise Introduction*. Princeton University Press, 2011.

[13] Songting Luo and Jianliang Qian. Fast sweeping methods for factored anisotropic eikonal equations: multiplicative and additive factors. *Journal of Scientific Computing*, 52(2):360–382, 2012.

[14] Lev Semenovich Pontryagin. *Mathematical theory of optimal processes*. CRC Press, 1987.

[15] Elisabeth Rouy and Agnès Tourin. A viscosity solutions approach to shape-from-shading. *SIAM Journal on Numerical Analysis*, 29(3):867–884, 1992.

[16] Walter Rudin et al. *Principles of mathematical analysis*, volume 3. McGraw-Hill New York, 1964.

[17] James A Sethian. A fast marching level set method for monotonically advancing fronts. *Proceedings of the National Academy of Sciences*, 93(4):1591–1595, 1996.

[18] James A Sethian and Alexander Vladimirsky. Ordered upwind methods for static hamilton–jacobi equations: Theory and algorithms. *SIAM Journal on Numerical Analysis*, 41(1):325–363, 2003.

[19] James Albert Sethian. *Level set methods and fast marching methods: evolving interfaces in computational geometry, fluid mechanics, computer vision, and materials science*, volume 3. Cambridge university press, 1999.

[20] Hector J Sussmann and Jan C Willems. 300 years of optimal control: from the brachystochrone to the maximum principle. *IEEE Control Systems Magazine*, 17(3):32–44, 1997.

[21] Eleuterio F Toro. *Riemann solvers and numerical methods for fluid dynamics: a practical introduction*. Springer Science & Business Media, 2013.

[22] Eran Treister and Eldad Haber. A fast marching algorithm for the factored eikonal equation. *Journal of Computational Physics*, 324:210–225, 2016.

[23] John N Tsitsiklis. Efficient algorithms for globally optimal trajectories. *IEEE Transactions on Automatic Control*, 40(9):1528–1538, 1995.

Appendix

```
#include <cmath>
#include <cstdio>
#include <cstdlib>
#include <iostream>
#include <memory.h>
#include <limits.h>

#define DBL_MAX 1.7976931348623158e+308
#define pi 3.14159265359

#include "MinHeap.h"
#include "SpeedFunction.h"
#include "InitialCondition.h"
#include "ReEvaluateNeighbor.h"

using namespace std;
//::::::::::::::::::::::::::::::::::::::::::::::::::::::::
void fastMarching(int N, double *U)
{
    double h = 1/((double)N -3);
    int n = N*N;

    // Generate the speed function on each node
    double *F = new double [n];
    speedData(N, h, F);

    // Generate the value function on each node
    for(int i=0; i<n; i++)
    {
        U[i] = DBL_MAX;     // Set the initial values on nodes as
    infinity
    }

    // Generate the status indicator on each node
    int *status = new int [n];     // 0:Far  1:Consider  2:Accept
    for(int i=0; i<n; i++)
    {
        status[i] = 0;     // Set the initial status as "Far"
    }
    double obc = 0.25;

    // Initialize the boundary:
    initialBoundary(N, U, status, obc);

    // Initialize the heap
    struct MinHeap* minHeap = createMinHeap(n);
    minHeap->hsize = 0;

    //int K = N*N/2;
    minHeap->array[0] = newMinHeapNode(N+1, 0);
    minHeap->pos[N+1] = 0;
    minHeap->hsize = 1;

    // MAIN ALGORITHM
    double Uh,Uv = 0;
    int minnode = 0;

    while(!isEmpty(minHeap)){

        // Step one: Extract the root node of the heap and set its status as Accept
        struct MinHeapNode* minNode = extractMin( minHeap );
        minnode = minNode->node;
        status[minnode] = 2;

        // Step two: Re-evaluate the neighbors of the min node
        reEvaluateNeighbor(minnode+1, N, h, U, status, minHeap, F[minnode+1]);

        reEvaluateNeighbor(minnode-1, N, h, U, status, minHeap, F[minnode-1]);

        reEvaluateNeighbor(minnode+N, N, h, U, status, minHeap, F[
    minnode+N]);

        reEvaluateNeighbor(minnode-N, N, h, U, status, minHeap, F[
    minnode-N]);

    }     // End of while
}
```

Listing 9.1 C++ code

```
#include <cmath>
#include <cstdio>
#include <cstdlib>
#include <iostream>
#include <memory.h>
#include <limits.h>

#define DBL_MAX 1.7976931348623158e+308
#define pi 3.14159265359

using namespace std;
//::::::::::::::::::::::::::::::::::::::::::::::::::::::
double reEvaluate(double h, double U_node,
double Uh, double Uv, double F_node)
{
    double q = h/F_node;
    double U_new = U_node;
    double V = 0;
    double U1,U2= 0;

    U1 = Uh + Uv;
    U2 = Uh - Uv;

    if ( fabs(U2) <= q ){
        V = 0.5*( U1 + sqrt( 2*q*q - U2*U2 ) );
        if(V < U_node){
            U_new = V;
        }
    }else{
        V = min(Uh, Uv) + q;
        if( V < U_node ){
            U_new = V;
        }
    }
    return(U_new);
}
//::::::::::::::::::::::::::::::::::::::::::::::::::::::
void reEvaluateNeighbor(int node, int N,
double h, double *U, int *status,
struct MinHeap *minHeap, double F_node){

    if( status[node] != 2 ){

        double Uh = min( U[node-1], U[node+1] );
        double Uv = min( U[node-N], U[node+N] );
        U[node] = reEvaluate(h, U[node], Uh, Uv, F_node);

        if( status[node] == 0){

            minHeap->array[minHeap->hsize] = newMinHeapNode(node,
   DBL_MAX);
            minHeap->pos[node] = minHeap->hsize;
            minHeap->hsize = minHeap->hsize + 1;

            decreaseKey(minHeap, node, U[node]);

            status[node] = 1;

        }else{

            decreaseKey(minHeap, node, U[node]);
        }
    }     // End of neighbor
}
```

Listing 9.2 Original FMM

```
#include <cmath>
#include <cstdio>
#include <cstdlib>
#include <iostream>
#include <memory.h>
#include <limits.h>

#define DBL_MAX 1.7976931348623158e+308
#define pi 3.14159265359

using namespace std;
//::::::::::::::::::::::::::::::::::::::::::::::::::::
double reEvaluate(int N, double h, int node,
double *U, int *status, double *T, double *Tx, double *Ty,
                  double *T1, double F_node){

    double q = h/F_node;
    double U_new = U[node];
    double V,V1,V2 = 0;
    double Uh,Uv= 0;

    double ew,ns = 0;
    if( U[node-1] < U[node+1] ){
        Uh = T1[node-1];    ew = 1;
    }else{
        Uh = T1[node+1];    ew = -1;
    }
    if( U[node-N] < U[node+N] ){
        Uv = T1[node-N];    ns = 1;
    }else{
        Uv = T1[node+N];    ns = -1;
    }

    double U1 = min( U[node-1], U[node+1] );
    double U2 = min( U[node-N], U[node+N] );

    if( U[node] > U1 || U[node] > U2  ){

        double A = 2;
        double B = 2*( ew*h*Tx[node] + ns*h*Ty[node] - Uh - Uv );
        double C = h*h*( Tx[node]*Tx[node] + Ty[node]*Ty[node] ) - q*q
                     - 2*h*( ew*Tx[node]*Uh + ns*Ty[node]*Uv ) + Uh*Uh + Uv*Uv;

        double Delta = B*B - 4*A*C;

        if( Delta >= 0 ){// If solution exists, then we must use it

            V = (  sqrt(Delta) - B )/(2*A);

            if( V + T[node] < U[node] ){
                T1[node] = V;
                U_new = V + T[node];
            }

        }else{//If solution does not exists, then use one-side update

            V1 = Uh + q - ew*h*Tx[node];
            V2 = Uv + q - ns*h*Ty[node];

            V = min( V1, V2 );
            if( V + T[node] < U[node] ){
                T1[node] = V;
                U_new = V + T[node];
            }
        }
    }
    return(U_new);
}
//::::::::::::::::::::::::::::::::::::::::::::::::::::::
void reEvaluateNeighbor(int node, int N,
double h, double *U, double *T, double *Tx, double *Ty, double *T1,
                        int *status, struct MinHeap *minHeap, double
   F_node){

    if( status[node] != 2 ){

        U[node] = reEvaluate(N, h, node, U, status, T, Tx, Ty, T1,
   F_node);

        if( status[node] == 0){

            minHeap->array[minHeap->hsize] = newMinHeapNode(node,
   DBL_MAX);
            minHeap->pos[node] = minHeap->hsize;
            minHeap->hsize = minHeap->hsize + 1;

            decreaseKey(minHeap, node, U[node]);
            status[node] = 1;
        }else{

            decreaseKey(minHeap, node, U[node]);
        }
    }// End of updating node
}
```

Listing 9.3 Factored FMM

Acknowledgements

I want to thank Prof. Alexander Vladimirsky for his detailed instructions on the background of this project and ideas of using the Factoring technique. His suggestions on how to build up the factoring function really helps me deal with this difficult part. I also want to thank him for his insightful comments on this graduation project thesis;

I want to thank Prof. Lei Zhang for his kind advice on how to organize my thesis paper and how to make a good presentation. His suggestions on the order of the materials in my presentation really helps me; I also need to thank him for his kindful patience and considerate guidance;

I also need to thank my parents and my sister, who have always been supporting me of my choices; Without their support, I am more encouraged to fulfill my dream of studying mathematics and doing mathematical research.

Thank my friends in SJTU and in CAM of Cornell University who have helped me solve the difficult parts of this project; Thank all the teachers who have helped me in my study and research. Without these kind people, I will never be able to finish this thesis.